Master's Thesis in Systems and Control Engineering 2016

Lilly Eirin Eikehaug

Development of Prototype for Registration of Road Data

University College of Southeast Norway
Faculty of Technology

University College of Southeast Norway
Faculty of Technology
Department of Electrical Engineering, Information Technology and Cybernetics
Kjølnes ring 56
3918 Porsgrunn

http://www.usn.no

Abstract

The Norwegian Public Roads Administration is responsible for the state and county public roads in Norway. This involves planning, construction and operation of the road network. NPRA manages the National Road Data Bank, which is a database that consist of information about the state, county and private roads.

When there is need for control or registration of new road objects, a registration application is used. However, the general registration applications are complex and usually handled by employees mainly working in the registration field. Based on this it is of interest to have a simplified application regarding a smaller field, such as road objects concerning water. The road object in focus for this thesis is the culvert. Culverts are installed across the roads and serves the purpose of leading water. They are crucial to handle large amount of rainfall along the road network.

The main goal of the thesis is to analyze, design and develop an application for road registration. The graphical user interface of the application is designed based on the need for a simplified application that could be used in fieldwork. Making a “tablet-friendly” application, with bigger fonts, buttons and fewer distractions.

The culvert registration tool retrieves road object information from NRDB by using the NRDB API, information such as culvert properties, existing culverts, road references etc. The registration procedure starts by presenting required and conditional properties. Later in the registration procedure, more properties of other categories can be added if needed, as well as associated objects of the culvert. The application uses an interactive map to set the geographical coordinates of the culvert and to display existing culverts for monitoring.

This thesis presents the research phase leading up to the development of the application and the documentation of the analyzing, design and development of the prototype for a registration application concerning culverts. A user manual for the application has been developed to show the program content and functionality in a systematic way.

Contents

Preface

This thesis is written during the spring semester of 2016, as the ending project of the master's degree in Systems and Control Engineering at the University College of Southeast Norway (HSN).

I would like to thank my supervisor Hans-Petter Halvorsen for weekly guidance throughout the project and external supervisors from Norwegian Public Roads Administration for providing the thesis, giving valuable input and for giving an informative demonstration of how the road registration process works in practice today.

My gratitude goes to those who have supported and encouraged me throughout my studies. My family, my study group; Alexander and Donika and other friends; Ingvild, Mariann, Tiril and Tone.

Porsgrunn, 3. June 2016

Lilly Eirin Eikehaug

1 Introduction

The main goal of this project is to develop a prototype to register data to the National Road Data Bank, NRDB (In Norwegian: NVDB). The NRDB is a database consisting of information about the road network. The data connected to the road network is road objects such as road signs, railing, tunnels, etc., events along the roads such as traffic accidents, avalanches, etc., and road statistics such as amount of traffic in areas, traffic noise etc.

Norwegian roads are under constant development, regarding both maintenance and expansion. During road development, entrepreneurs are responsible for taking road object measurements. The measurement data are mapped by surveyors or collected from the entrepreneurs' project-or building tools. The road object geometry and relevant information is handed over by the entrepreneurs for registration.

Under some circumstances there is still a necessity for the Norwegian Public Roads Administration (N[1]: Vegvesenet) to travel to the designated road to control and register the data. Employees that specialize in registration and measurements of road objects are responsible for this work. The equipment and software used in this registration process are complex due to the wide amount of road objects.

Based on this, there is an interest and need for a simplified registration application which is narrowed down to certain areas. Ideally, a simple and efficient application that employees, who do not work directly with registration, will be able to operate. In cooperation with the Norwegian Public Roads Administration, the decided road object for registration in the prototype is culverts.

The focus of this work is to analyze, design and develop a registration application for culverts, aiming to make it user-friendly and efficient to operate. This report consist of two main parts, the first part presenting the background and research phase and the second part, the application development.

[1] During this thesis, "N" and "E" will state the translation to Norwegian and English.

Part I : Background & Theory

The following chapters will present the research part of the thesis. Starting with an overview of the thesis description and main problem, as well as the decisions made when narrowing down the thesis, based on cooperation with supervisors and staff from NPRA.

This part will present background information about the NRDB, its area of use and related application tools. There will be a chapter with general information about the registration process today and the chosen road object of focus for the application. The ending of part 1 consist of information about geographic localization, including overviews of Global Navigation Satellite System (GNSS), the position correction service CPOS and the National Road Reference System.

2 Background

This background chapter will introduce the problem description provided by NPRA, which is a very general description. This chapter will describe the redefining and adjustments of the task, due to areas of interest and equipment limitations.

2.1 General Problem Description

The Norwegian Public Roads Administration provided the problem description. NPRA is responsible for the planning, construction and operation of the state and county road networks in Norway. The task description can be found in Appendix A.

The task description specifies; analyze, design and develop a prototype for an application that can register typical road data mounted on and along the roads. The registration of road data application should communicate with the NRDB API for reading and writing of data, as shown in Figure 2-1. The use of GPS location is also of interest. Challenges may be that the system should work under all kind of weather conditions and the importance of accurate GPS localization.

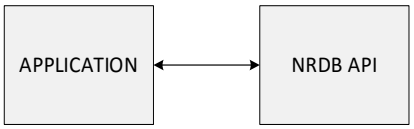


Figure 2-1: Communication between the application and the NRDB API

2.2 Focus Areas and Task Limitations

After dialogue with NPRA, the main road object in focus for this thesis is culverts. It is of interest to have an application that is limited down to only a few road objects as the larger registration application can be rather complex. The focus area was narrowed down to creating a Graphical User Interface (GUI) that could easily be operated in fieldwork by employees that do not necessarily work in the field of road registration.

The data needed for the application can be retrieved from NRDB by using the reading-API, which is open to the public. However, it's not possible to access the writing part of NRDB. A developer tool for the writing-API will be available to establish connection.

As the GNSS equipment used in road registration today is expensive and therefore unavailable during this thesis, the GNSS part will belong to the research phase. The positioning of road objects will be handled with other methods within the application.

3 National Road Data Bank (NRDB)

This chapter will give an overview of the Norwegian National Road Data Bank, which is an open database with road data free for the public to access and use. There will be an overview of what the NRDB consist of, licensing procedure and relevant user groups. This chapter will also present a widely used web application with the possibility to explore NRDB data and pin road objects to a map, introduce the data catalog “Datakatalogen” and the NRDB REST-API.

3.1 NRDB

NRDB (N: NVDB) is the Norwegian National Road Data Bank managed by the Norwegian Public Road Administration (NPRA), which is a database consisting of information about roads that are owned by the state, county or private. [1]

The database consist of [1]:

- Road network with geometry and topology used in solutions for maps and route calculations for web applications
- Overview of equipment and road objects such as traffic signs, tunnels, bridges etc.
- Overview of airports, ferries, subways etc.
- Details about traffic accidents and amount of traffic
- Data used for traffic models and noise estimation

3.1.1 License

The information stored in the NRDB is available through the NLOD (N: Norsk lisens for offentlige data) which is the Norwegian license for public data. Which allows you to use the information freely as long as it is referenced back to its origin source. It also states that the database might have errors and deficiencies. [1, 2]

When using data from the NRDB, the following should be stated in the paper: “Consists of data by Norwegian license for public data (NLOD) made available by Norwegian Public Roads Administration. “ [1]

3.1.2 User Groups

The NRDB accessibility divides into three main groups: internal, external and public. The information in the database is accessible for the public users to access and use, while internal and external users have additional authorization to alter and register data in the database. The internal users are employees of the NPRA, while external users have

occupations with a connection to the NPRA, such as government agencies, police, National Bureau of Statistics to mention some. [3]

3.1.3 Purpose

To sum up, the purpose of the NRDB is to present information in a strategic and efficient way to use in [3]:

- Development and construction of new roads
- Management and maintenance of existing roads
- Information availability for users; internal, external and public

3.2 Road Map Application

The Norwegian Public Roads Administration has developed an application called Roadmaps (N: Vegkart) where the users can search the NRDB in the internet browser. The user can simply navigate the map by using the pointer, or by searching for specific areas in the search field.

The application consists of a catalog with varying road information, where the user can navigate to the desired option. The search field is used to search for specific road data, as an example: tollbooths, traffic accidents, speed limits. Figure 3-1 shows the interface of the road map web application.

Figure 3-1: The Interface of the Roadmap web application

The road map is available for the public on: https://www.vegvesen.no/vegkart/vegkart/

The data used for the road map application retrieves information from the NRDB API (launched 27/02/2013). Developers are able to use the API to program own information requests. Both the road map and the NRDB API is still under a lot of updating and development. [4]

3.3 Data Catalog

The NRDB data catalog (N: Datakatalogen), is a registry of all the road objects within NRDB, including all their associated properties with listings of possible defined input values or the format of the input (string, number, date). The data catalog also shows how the different road objects are associated. [5] The newest updated versions of the data catalog are available from: http://tfprod1.sintef.no/datakatalog/

Figure 3-2 shows the data catalog application Datakatalog 2.05, where the road object type culvert (N: Stikkrenne/Kulvert) is chosen under the road object category drainage (N: Drenering), opening up the property type (N: Egenskapstype) ‘area of use’ (N: Bruksområde), and listing its possible inputs (N: Tillate verdier).

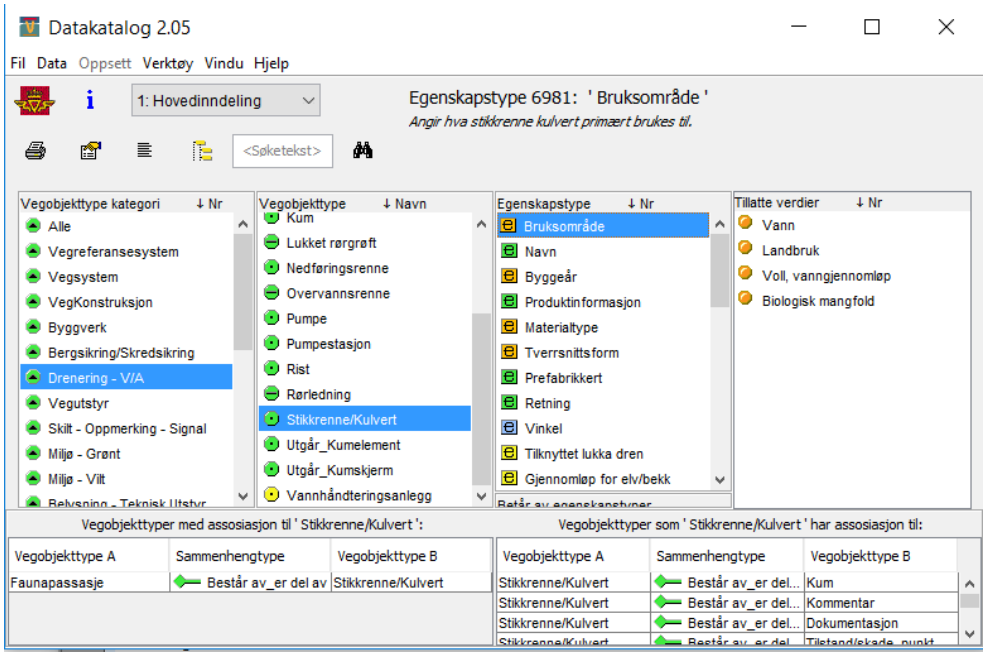


Figure 3-2: The NRDB Data Catalog displaying possible input values for the culvert property ‘area of use’

3.4 NRDB API

The NRDB Application Programming Interface (API) is an interface that allows the user (programmer) to communicate with the databank managed by the NPRA. The data is in XML (Extensible Markup Language) or JSON (JavaScript Object Notation) format. The database consists of 250 different object types, with total 16 million objects. All the objects have a belonging coordinate for display purposes. [6]

3.4.1 NRDB Reading-API

The NRDB reading-API is open for the public to use.

NVDB rest-API BETA documentation is available on:
https://www.vegvesen.no/nvdb/api/dokumentasjon/

3.4.2 NRDB Writing-API and Developer Tool

Unlike the reading-API, the writing-API for the NRDB is not available to the public. To add new and alter road objects in the NRDB, the developer is required to have a user in the Norwegian Public Roads Administration and have data rights to the specific road objects, areas and road categories under development. [7]

NPRA launched a developer tool for the writing-API March 2016. The developer tool is available using docker-hub. It makes it possible to create property sets for road objects and have them validated to the NRDB. It is mirroring the infrastructure of the NPRA and the NRDB database, making the writing procedure similar to the actual process.

4 Road Registration Demonstration

To be able to develop a realistic solution for the road object registration application, it is necessary to get a clear understanding of how the registration process is working today. This chapter will give a system overview of the road registration procedure, introduce the relevant GNSS receiver and the road reference application **Vegviseren**.

4.1 Road Registration System Overview

Figure 4-1 presents a detailed overview of the registration system with hardware and software components.

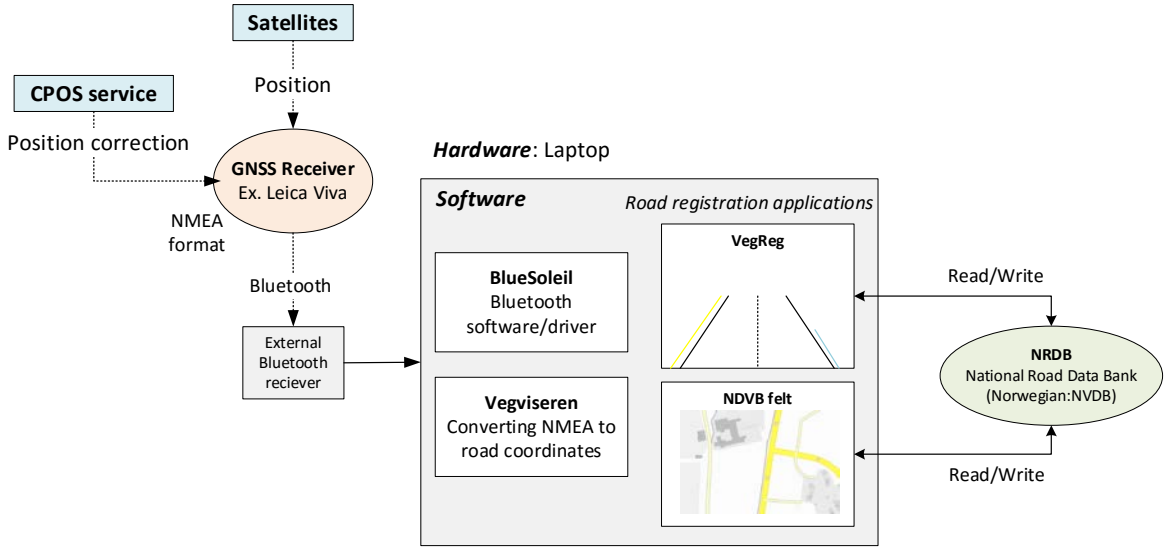


Figure 4-1: Overview of the road registration system

Starting from the left-hand-side is the GNSS Receiver, receiving position based on satellites and position correction data from CPOS service. Retrieving GNSS data such as position and other specifications regarding accuracy using NMEA-messages. The GNSS receiver is a wireless device, which transmits the data using Bluetooth to an external Bluetooth receiver attached to the computer used for road registration.

The computer used for road registration is running several applications which are interconnected to perform the main tasks, which is to monitor and register road data. To receive the GNSS data using Bluetooth communication, the application BlueSoleil is used, while Vegviseren is responsible for the conversion from NMEA-format to appropriate road references, which can be used in the main registration programs. The main registration programs used is **VegReg** and **NDVB felt** which serves the same purpose, which is reading and writing to the NRDB, but holds different interfaces with its own pros and cons, making

both programs necessary. More about the road registration vehicle, the object registration, GNSS receiver and **Vegviseren** will be discussed in the following chapters.

4.2 Road Registration Vehicle

As part of the research, Norwegian Public Roads Administration employee, Kristian André Gallis who drives a road registration vehicle in the Vestfold region gave an introduction and demonstration of the equipment and programs used in the process of road object registration. Figure 4-2 shows the road registration vehicle with the GNSS receiver attached on the right-hand side of the vehicle.

Figure 4-2: The road registration vehicle with the attached GNSS receiver

The vehicle is equipped with a laptop running the registration software, and an external screen for additional passenger/assistant. Each region typically has one employee with main responsibility for the general registration of road objects within the region. There is also subgroups with responsibilities for specific sets of road objects, such as road signs.

4.3 Object Registration

Specified for field registration, a windows surface tablet running **VegReg** and **NVDB felt**. *Figure 4-3* shows the Windows tablet attached to the GNSS pole, operated by using a tablet pen. The need for a pen is due to lack of a tablet-adapted application, and the complexity of the program itself.

On the right hand side is a long-range Bluetooth receiver, which receives the location data from the GNSS attached to the top of the pole.

Figure 4-3: Windows surface tablet with registration software

Two programs are currently in use to register the road objects, **VegReg** and **NVDB Felt**, both using the old API. Both programs are necessary due to the lack of functionality in the two. One of the programs displays a straight road in perspective, while the other application displays an overlooking map of the road. The first program makes it possible to run distance tracking on several objects and the time, while the other program gives a better display of road objects such as roundabouts and so on.

4.4 GNSS Receiver

The GNSS receiver is attached to the side of the vehicle to retrieve a precise location of where the vehicle is located when working on the roads. National Road Public Administration currently have contracts with Leica and Trimble, which are their main equipment provider. The equipment used during this demonstration was a Leica GNSS receiver shown in Figure 4-4.

The GNSS receiver is transferring its position data using Bluetooth. The registration computer has a USB-connected Bluetooth antenna inside the car. Using an external antenna is necessary for a clear signal to travel inside the vehicle. BlueSoleil is the Bluetooth driver/software used to establish Bluetooth connection, before the data is ready for use in the registration software.

Figure 4-4: Leica GNSS Receiver

The GNSS receiver is used as a handheld device when there is a need for more accurate geometry of an object, placing the pole vertically upon the desired point using an integrated leveler. The GNSS is using CPOS service (more about CPOS in Chapter 6.3) for position correction to achieve centimetre accuracy. The GNSS receiver is battery driven and is able to last for an entire workday.

4.5 VegViseren

VegViseren is an application used when working along the roads. It provides continuous positioning, such as coordinates, road reference and road names. [8]

The GNSS receiver gets the current position based on satellite distance. The position data retrieved is in NMEA-format, and is converted into UTM33/WGS84. **VegViseren** calculates the distance to the closest road by reading from the database containing road geometry. The road reference and GPS-data is displayed in the **VegViseren** application, and can be used in additional road registration applications. [9]

In the current registration process, **VegViseren** converts data from NMEA-format to road reference data. The application output is used in the road registration applications such as **VegReg** and **NVDB-felt**.

5 The Road Object in Focus: Culvert

This chapter gives a general description of the road object culvert, which is the road object in focus for this thesis. The registration tool discussed in part 2 of this thesis is created specifically for registration of new culverts or monitoring of existing culverts. The information in this chapter is based on the product specification in Appendix B.

For the road system to handle large amounts of rainfall, installation of culverts across the roads is crucial. Concept illustration of a culvert in shown in Figure 5-1.

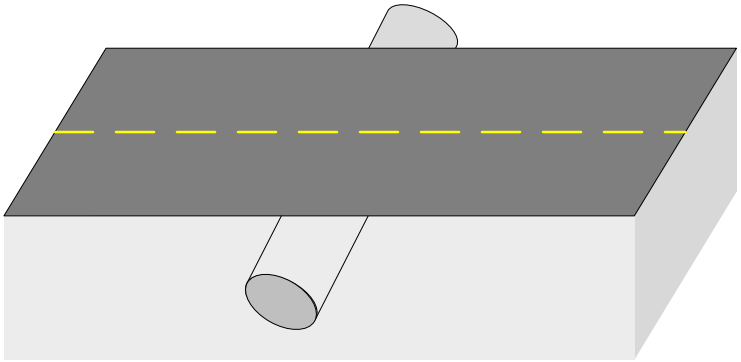


Figure 5-1: Illustration of a culvert installed across the road

The four main areas of use is:

- passage of water across the road
- passage of water through embankment
- passage of water underneath a road associated with agriculture
- passage of water that prevents the road to limit biodiversity

Accurate data of the culvert is necessary for planning, maintenance and operation. Its precise position is important to be able to locate the culvert when there is challenging weather conditions such as heavy snowfall. The position accuracy is set to 20 cm. The culvert position displays as a point in the road network.

The culvert has a maximum opening of 2.5 meter, with open inlet and/or outlet. The in-and outlet might consist of manholes or support shields. It can be made of various material types such as concrete, steel, plastic, natural stone, tree or eternit and have different shapes like circular, rectangular, ellipse, flat-bottomed etc.

The direction of the culvert is given as a number from 1-12 representing the clock direction. The arrow is indicating the flow direction, shown in Figure 5-2.

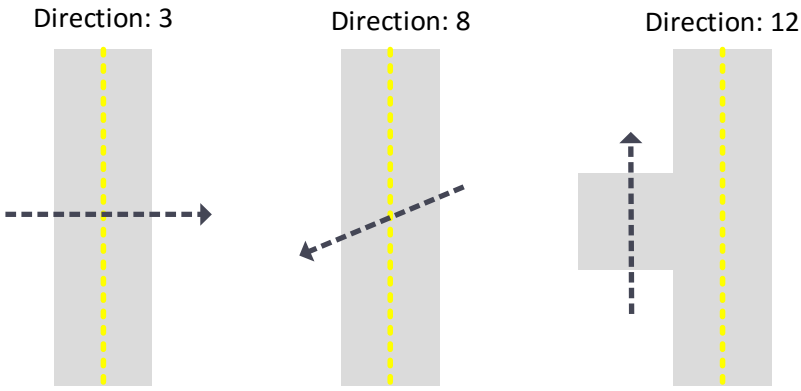


Figure 5-2: Culvert direction, showing direction 3, 8 and 12

Road objects associated with the culverts are (consist of/part of):

- Manhole
- Comment
- Documentation
- Condition/damage, point
- Condition/damage FU, point
- System object
- State level, culvert (test)

6 Geographic localization

One of the facing challenges due to detecting and registration of road data is the positioning accuracy of the object. It is essential that the GNSS receiver can provide localization of the object with reasonable accuracy. This chapter will present Global Navigation Satellite Systems (GNSS), geographic coordinate systems, the corrections service CPOS, NMEA-format for position data and finally an introduction to the national road reference system.

6.1 Global Navigation Satellite System (GNSS)

Figure 6-1 displays a GNSS system overview. The GNSS receiver calculates position data from GNSS Satellites and position correction data from services like CPOS. The correction data is calculated by using a base station with a known location.

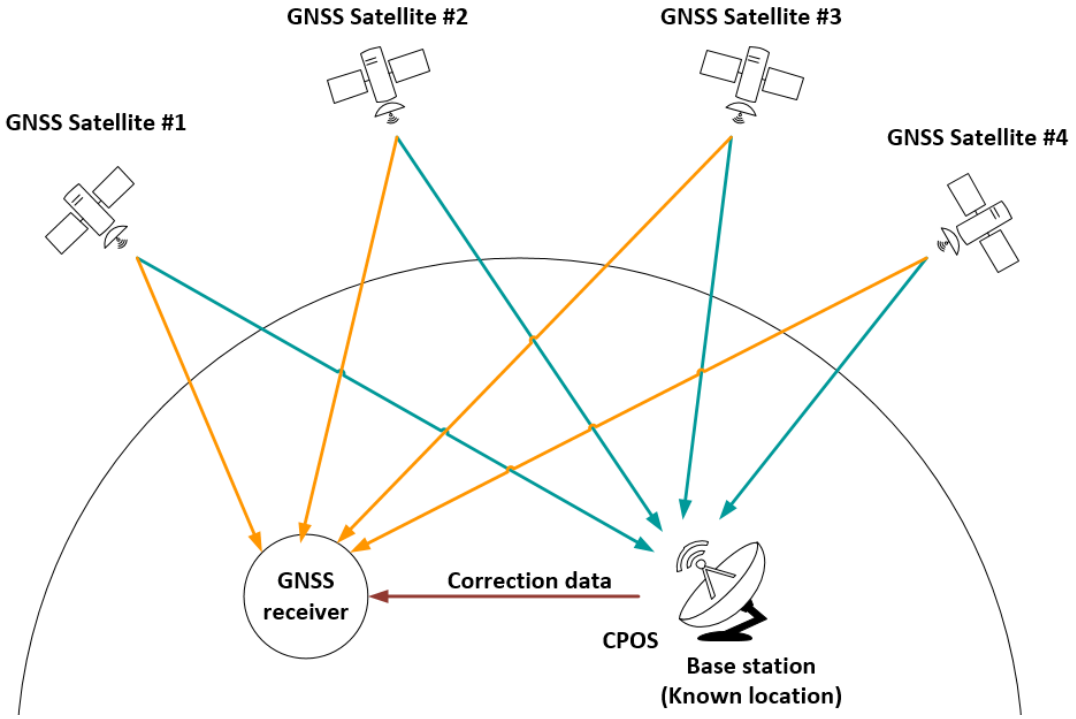


Figure 6-1: GNSS System overview displaying GNSS receiver, GNSS satellites and base station with known location

GNSS is based on mobile receivers, collecting signals from several satellites. To calculate the distance from the satellites to the receiver, TOA (Time of Arrival) principle is used, which is the time it takes for the satellite signal to arrive at the receiver. This signal consist of a time stamp and satellite positioning data. The receiving object calculates the position based on trilateration. [10, 11] To get navigation in 3D, three or more satellites have to be

in direct line of sight of the receiver. While the three satellites are used to track the position, a forth satellite is needed for time synchronization. [12]

Today it exists three operational GNSS systems; the US GPS, Russian GLONASS and Chinese BeiDou, while Japan, India and Europe (Galileo) is still developing their own systems. GPS and GLONASS satellites orbit earth with a height of approximately 20.000 km, while BeiDou and Galileo orbit higher, respectively at approximately 21.500 km and 23.000 km. [10]

6.1.1 Global Positioning System (GPS)

Global Positioning System (GPS) is one of the most used Global Navigation Satellite Systems (GNSS), operated by the U.S Air force. GPS requires having 24 operational satellites, orbiting earth at all times. In total, there is 31 satellites to ensure high operability. [11] The signal transmission is CDMA (Code Division Multiple Access) channel access method based.

6.1.2 Global Navigation Satellite System (GLONASS)

Global Navigation Satellite System (GLONASS) developed originally for military purposes by the Soviet Union. GLONASS requires having 24 operational satellites. In total there are 28 satellites orbiting earth at all times. FDMA (Frequency Division Multiple Access is used for signal transmission). [13]

6.2 Geographic Coordinate System

To specify a location on the surface of earth, a geographic coordinate system is helpful. The location is described by using numbers, symbols and letters. Latitude, longitude (*Figure 6-2*) and altitude is the most common choice of coordinates.

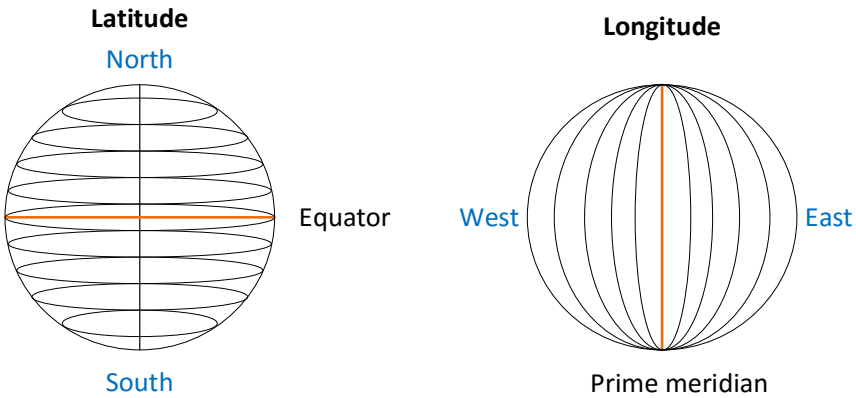


Figure 6-2: The principal of latitude and longitude [14]

Description of latitude, longitude and altitude:

- Latitude is described with lines parallel to equator (marked with the orange lines as shown in in Figure 6-2.
- Longitude is described with lines perpendicular to equator, east or west of the prime meridian (marked with orange lines as shown in Figure 6-2.
- Altitude is the height above sea level

The surface of the earth is irregular, and therefore not completely spherical. An earth ellipsoid is a mathematical description of the shape of the earth, used as reference in calculations. There has been several types of ellipsoids used for the approximations. [15]

When maps are created, a reference ellipsoid with given origin and orientation has to be chosen based on the purpose of the map, then the suitable mapping of the spherical coordinate system is added on, referred to as a geodetic datum. Datums are either global or local, representing the whole earth or a specific part of earth. The default datum for GPS is World Geodic System (WGS 84) (more about WGS in chapter 6.2.2). The datum will give location in latitude and longitude. Map projections are used to pin geographic positions on a flat surface map, it converts geodic coordinates to two-dimensional coordinates. Map projection and the datum pinned to a grid of location references make a grid system. Universal Transverse Mercator (UTM) is a commonly used map projection (further discussed in chapter 6.2.1) along with Military Grid Reference System (MGRS), the United States National Grid (USNG) and more. [14]

6.2.1 Universal Transverse Mercator Coordinate System

Universal Transverse Mercator (UTM) coordinates consists of 120 transverse Mercator map projections, one for each N/S hemisphere, and two for each of the UTM zones. The UTM coordinate system uses easting and northing to describe positions. The surface of the earth is lined up with 60 zones. The distance between the sectioning is 6° in longitude. [16] Figure 6-3 shows the dividing of the UTM zones with belonging zone numbers.

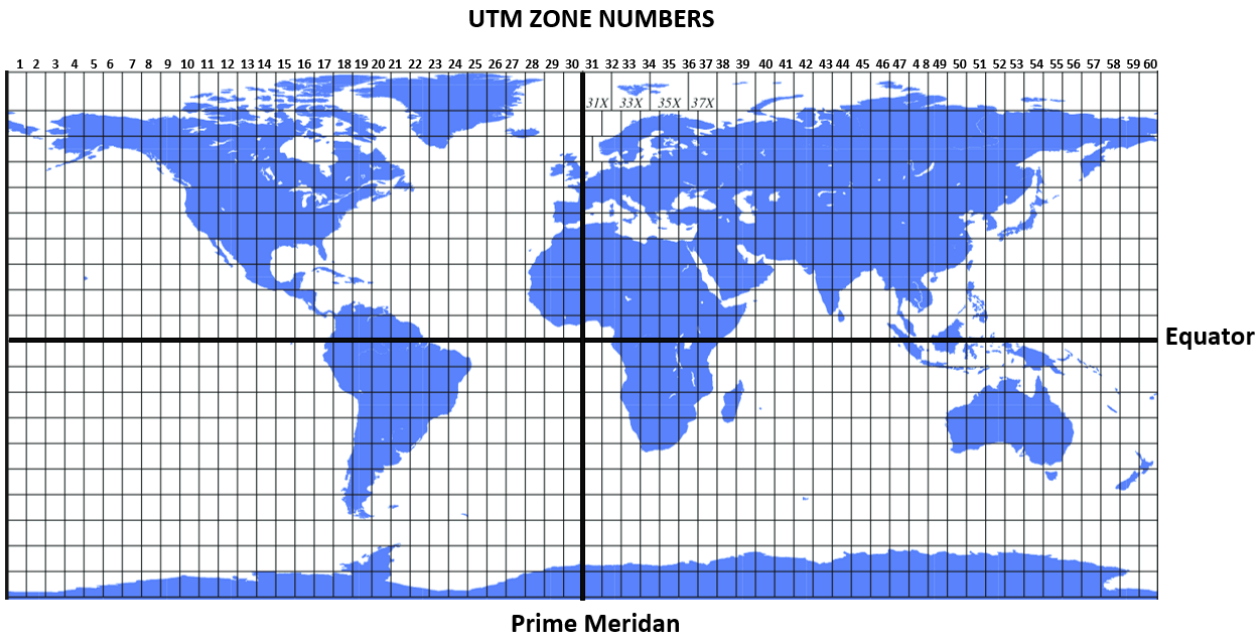


Figure 6-3: UTM zone numbers, figure based on reference [17]

Instead of giving location coordinates in latitude and longitude with °S and °N, the UTM coordinate system will describe the location according to the corresponding zone using northing and easting mN and mE. [17]

6.2.2 World Geodic System (WGS)

WGS84 is the last revision for the World Geodic System (WGS). WGS is a standard commonly used for operation and updating of GPS. It is a combination of a spheroidal reference surface giving raw altitude data, a standard coordinate system of the earth and gravitational equipotential surface, which describes the nominal sea level. The WGS84 origin is at the earth's center of mass, with an approximated error of 2 cm. [18, 19]

Smartphones and map applications, such as google earth uses the WGS84 to present geographical positioning, such as latitude and longitude.

6.3 CPOS

CPOS is short for centimeter position and is a service that makes it possible to achieve a position measurement with accuracy based on cm. CPOS is a great tool to obtain good accuracy when doing measurement work along the Norwegian roads. GPS and GLONASS users get correction data from CPOS, and continuously calculate its accurate position. CPOS is available using GSM- or GPRS modem.

The service creates a virtual reference station based on the permanent geodic stations spread across Norway and the current location of the user. [20]

6.4 NMEA Format

To determine the location of the desired road object, the application should handle GNSS-data on NMEA-format. National Marine Electronics Association (NMEA) is a standard format supported by all GPS devices. NMEA makes it possible to use different hardware and software solutions within the same system. The software developer is able to communicate with different brands of GPS when using NMEA-format, instead of having to create different interfaces to fit each type of GPS receiver, which saves both time and money. NMEA sends a segment of data using one sentence, concept shown in Figure 6-4. [21, 22]

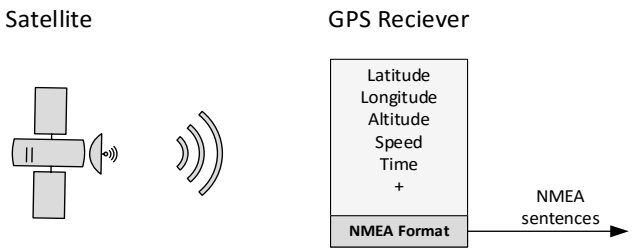


Figure 6-4: GPS data on NMEA format [22]

NMEA-data can be transmitted from the GNSS receiver by using USB, RS-232, Bluetooth, WI-FI, etc. communication interfaces. There is a wide range of NMEA messages with different abilities and properties. All NMEA messages starts with '$' and using ',' as separator. The NMEA message $GPGST is used when high precision data is required for the system. When determining the quality of a GPS coordinate, real-time metadata is useful, data such as PDOP for three-dimensional accuracy, how many satellites are in range, horizontal and vertical standard deviation values and correction methods. [21]

An example of a commonly used NEMA message structure is $GPGGA, an example of the output when using this message is:

$GPGGA, 122401.00, 4302.6002178, N, 06043.28554341, W, 3, 9, 1.00, 486.121, M, 0.10, 0000*40

In the according order, GP denotes that it is a GPS position, time stamp, latitude (DDMM.MMMMM format), N for north latitude, longitude (DDMM.MMMMM format), W for west longitude, quality indicator (5 different), amount of satellites in range, DHOP (Horizontal Dilution of Precision), antenna altitude, geoidal separation, unit of geoidal separation, age of correction, correction station ID, * checksum. [21]

NMEA data presents the GPS coordinate as degrees, minutes and decimal minutes (DDMM.MMMMM), which can be tricky to deal with. The latitude and longitude divides the value after two numbers to get the degrees, and the remaining numbers represents the minutes. Converting to decimal degree by dividing the minutes with 60. [21]

6.5 National Road Reference System

When mapping road objects, a reference system is necessary to be able to pin the information to the roads. The reference system consists of a network with peer-link connections.

The national road reference system is based on a network structure, known as the basic net (N: basisnettet). This network describes road directions and road crossings. The basic net main tasks are keeping track of how the roads are interconnected and how it's localized in the UTM-coordinate system. It also makes it possible to give road objects localization, independent from changeable properties. Changeable properties being county, road number etc.

The link represents a road distance and the peer represents the crossing of multiple roads or the end of the road (end of link). Peer and link connections are shown in Figure 6-5.

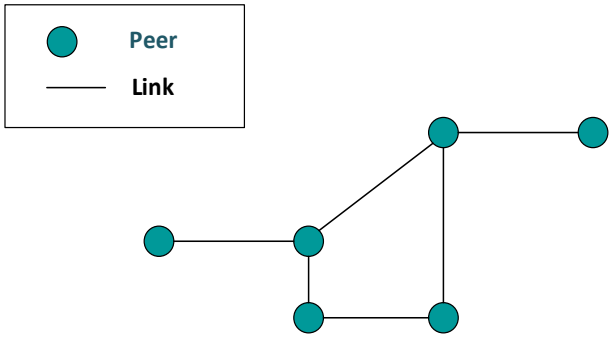


Figure 6-5: Peer-link connections

It is required that the links consist of information that describes if the link represents a road align, roadway or lane. To prevent having too short or too long links, the maximum length of a line is set to 2 km. The peers and links are localized with eastern and northern coordinates (UTM coordinate system), and height above/below sea level.

It is called indirect localization when road object are connected to its peer and retrieving its UTM coordinates, because the localization is happening through the basic net. It is also possible for road objects to have geometric coordinates directly connected to the UTM-system. [23]

6.5.1 The Meter Reference System

Within a county, each of the roads are divided into parcels that consist of an entire road route or part of a road route. A unique number within the county describes the parcel, which normally starts and ends at road crosses. The localization of a road section is described by meters within the parcel (from meter – to meter).

The meter reference system (N: Det metrerte referansesystemet) does not consist of peers and links. It is linear, and one-dimensional. It is line based and only refers to the distance from the start point of the road. The road reference is based on the meter system, where a point on the road is defined within a parcel, by giving the distance from the parcels start.

Figure 6-6 shows an example of a road reference, where the four first number represents the county number, municipality number. The next letter describes the road category which in this case is 'F' describing county road (N: Fylkesveg), next letter is road status which is 'V' describing that it is an existing road, next is the road number and HP describing the parcel, and lastly the meter values. [23]

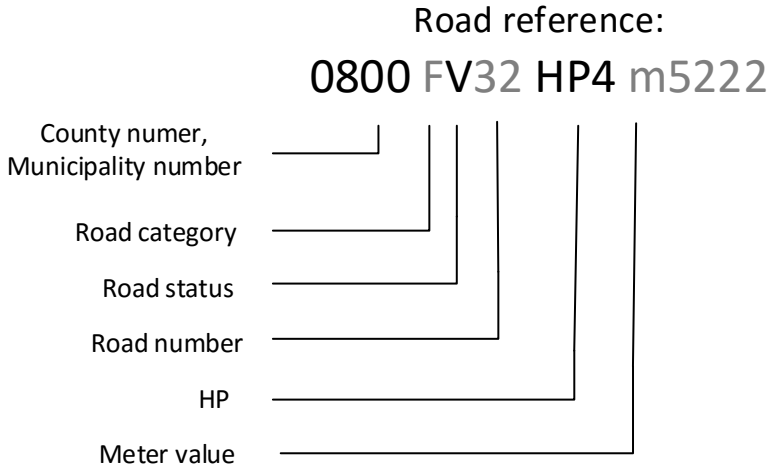
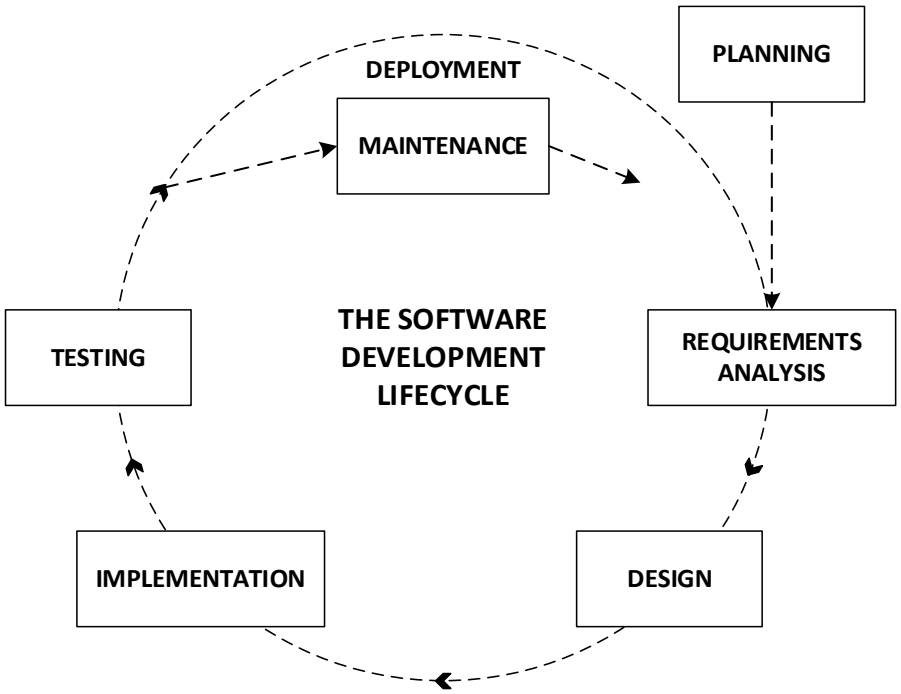


Figure 6-6: Example of a Road reference in Porsgrunn, Telemark

Part II : Application Development

The following chapters will focus on the development of the road object registration application, **Culvert Registration Tool**. Starting with the main concept and idea, the general development platform and design choices, as well as documentation of the planning process. This part will also present the graphical user interface and the functionality of the application with description and drafts of the code behind. There will be a closer look on how to operate an interactive map within the application, how to retrieve information from the National Road Data Bank using a NRDB reading-API and how to set up and write to its developer tool for testing NRDB writing-API.

Part 2, is based on the software development lifecycle shown in the illustration [24] below, going through the phases of planning, requirements and analysis, design, implementation, testing and deployment.



7 Application Planning

This chapter discusses the pre-preparation of the application development. Presenting the basic ideas and plans before the programming part of the project starts, planning techniques and programming language choices.

7.1 Main Idea

The core idea for the application is to make it minimalistic and user friendly, so it will be manageable by employees who are not particularly working in the field of road object registration. The programs used for general road registration today are complex systems, it is therefore of interest to have a simpler application regarding specific objects. After discussing with the Norwegian Public Roads Administration project supervisors, the choice of application development tool was to be specialized for culverts.

The system overview is shown in Figure 7-1. The road object registration application retrieves the position data from a GNSS. The application is reading information from-and writing new information to the NRDB.

Figure 7-1: System overview from GNSS to the application to NRDB

Based on equipment used today, Figure 7-2 explains the different components of a potential system. The GNSS receiver get position data from satellites and correction data from CPOS. The GNSS data is transferred to the application using a Bluetooth receiver and software/driver BlueSoleil. The application is using a map service, ex. Leaflet, for map display. The main application will be reading and writing to the NRDB.

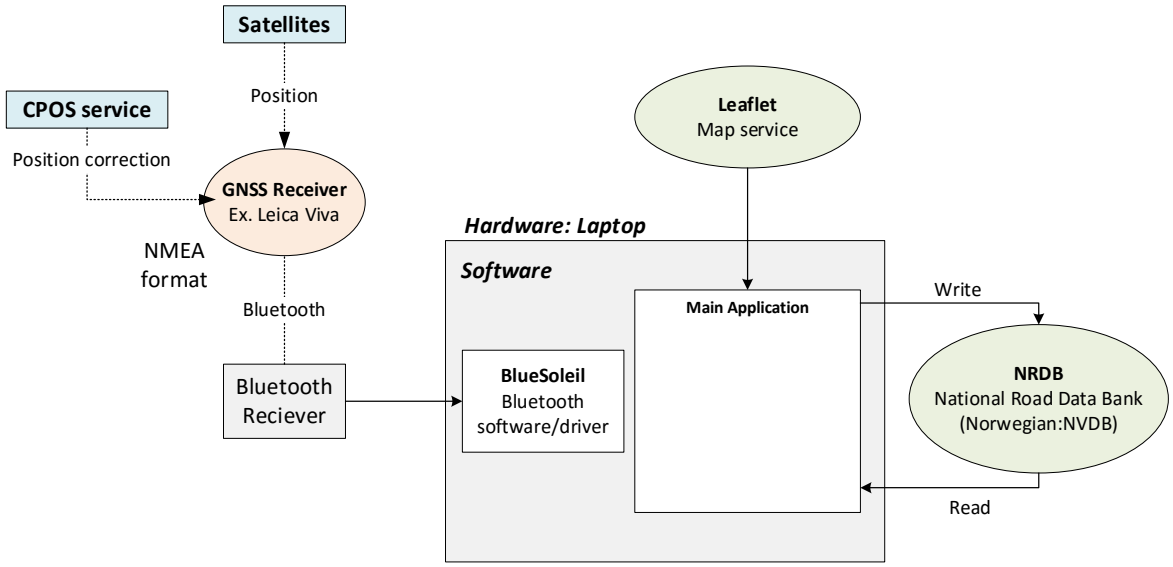


Figure 7-2: System overview of components in the registration of road data application

The GNSS receiver is not accessible for testing during the project, instead, the location of the road object will manually be set using an interactive map application within the program. The writing part of NRDB is closed to the public, but a NRDB writing tool developed for virtual saving of the data will be used.

7.2 Planning Phase

The following subchapters introduce drafts of the planning in the startup phase of the application development. The planning tools used is FURPS and Use Case Diagram, to map the main functionalities of the program.

7.2.1 FURPS+

FURPS is a planning tool for the functionality and requirements of a software program and is short for Functionality, Usability, Reliability, Performance and Supportability. The main focus is on the first parts; functionality and usability. [25] The FURPS is developed based on the task description and cooperation with external supervisors from NPRA.

F:

- Read API: Establish connection to NRDB. Retrieving relevant information from NRDB using NRDB reading-API
- Write API: Establish connection to NRDB. Write relevant information to NRDB using the NRDB writing-API (In this application using the development tool for writing to the NRDB)
- General Info: Requesting general object description
- Register: Register new culverts by selecting property values
- Monitor and Alter: Monitoring existing culverts and alter relevant information

U:

- Application Language: Norwegian
- Database: National Road Data Bank (NRDB)
- System: Windows OS
- Displaying description of culverts, properties for registration and existing culverts

R:

- Only running for short amounts of time
- Internet connection required

P: -

S: -

7.2.2 Use Case

The Use Case diagram is a visual presentation of the functionality described in the FURPS. The Use Case describes how the different parts are connected and how they are related to the users. Users being the operator and external applications. Figure 7-3 shows the Use Case diagram for the culvert registration tool application.

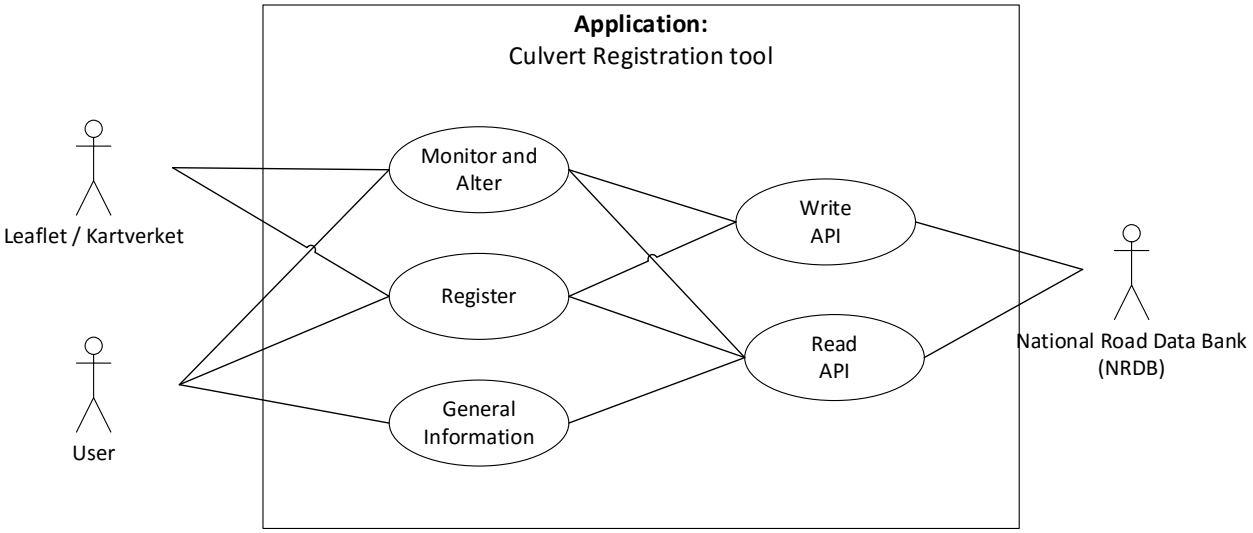


Figure 7-3: Use Case diagram for the culvert registration tool

The *User* is operators with authorization to read and write information to and from NRDB. *User* is connected to *General Information* which displays the description of the road object, *Register* is where it's possible to register geographical coordinates and properties of new culverts and *Monitor and Alter is* where it's possible to view and alter existing culverts within the NRDB. The external map service *Leaflet* is used in *Register* and *Monitor and Alter* for map displaying.

Monitor and Alter, *Register* and *General Information* are connected to *Read API*, which reads road information from NRDB using the NRDB reading-API. *General Information* retrieves general road object description. *Register* retrieves road object properties, associated objects and possible property inputs. *Monitor and Alter* retrieves information about existing culverts.

Monitor and Alter and *Register* are connected to *Write API*, which writes to the NRDB using the NRDB writing-API. *Write API* and *Read API* are connected to the National Road Data Bank (NRDB),

7.3 Programming Environment

The following subchapter introduces the choices of programming language and tools to develop the application. The chosen GUI framework is Windows Presentation Foundation (WPF) using XAML and C# language. The Integrated Development Environment (IDE) chosen is Visual Studio.

7.3.1 Programming Tool

The program used to develop the application for road registration is Visual Studio (2015) using the programming language C#, due to prior knowledge of the program and language. Visual Studio is also the preferred IDE (Integrated Development Environment) for WPF. [26]

7.3.2 WPF

WPF is short for Windows Presentation Foundation and is a Microsoft GUI framework. It is a combination of XAML mark-up and .NET language such as C# or Visual Basic. WPF is used with .NET framework. WPF was particularly chosen because of its flexibility when it comes to design options, as WPF is built from scratch, it is not limited to using windows controls. [27, 28]

7.3.3 User Control

When developing the application interface, UserControl is used for all the “pages” within the application. Instead of using a *Window* or a *Page*, the root element is a UserControl. WPF User Control is an expansion in customization for the design and functionality. Each of the pages can consist of its own set of controls adapted to the application requirements. [29]

8 Graphical User Interface (GUI)

The graphical user interface is an essential part of the application development, as the focus is on creating an application that is specially adapted to a tablet. A tablet friendly application that should be possible to operate by using finger touch.

When maneuvering an application used in field work, it should be intuitive and simple to guide through the application in an efficient way. There should be limited content on the pages and bigger fonts and components.

The graphical user interface in the application uses the design of Norwegian Public Roads Administrations webpages as a base, to provide familiarity and simplicity for its users. Figure 8-1 shows the homepage of vegvesen.no.

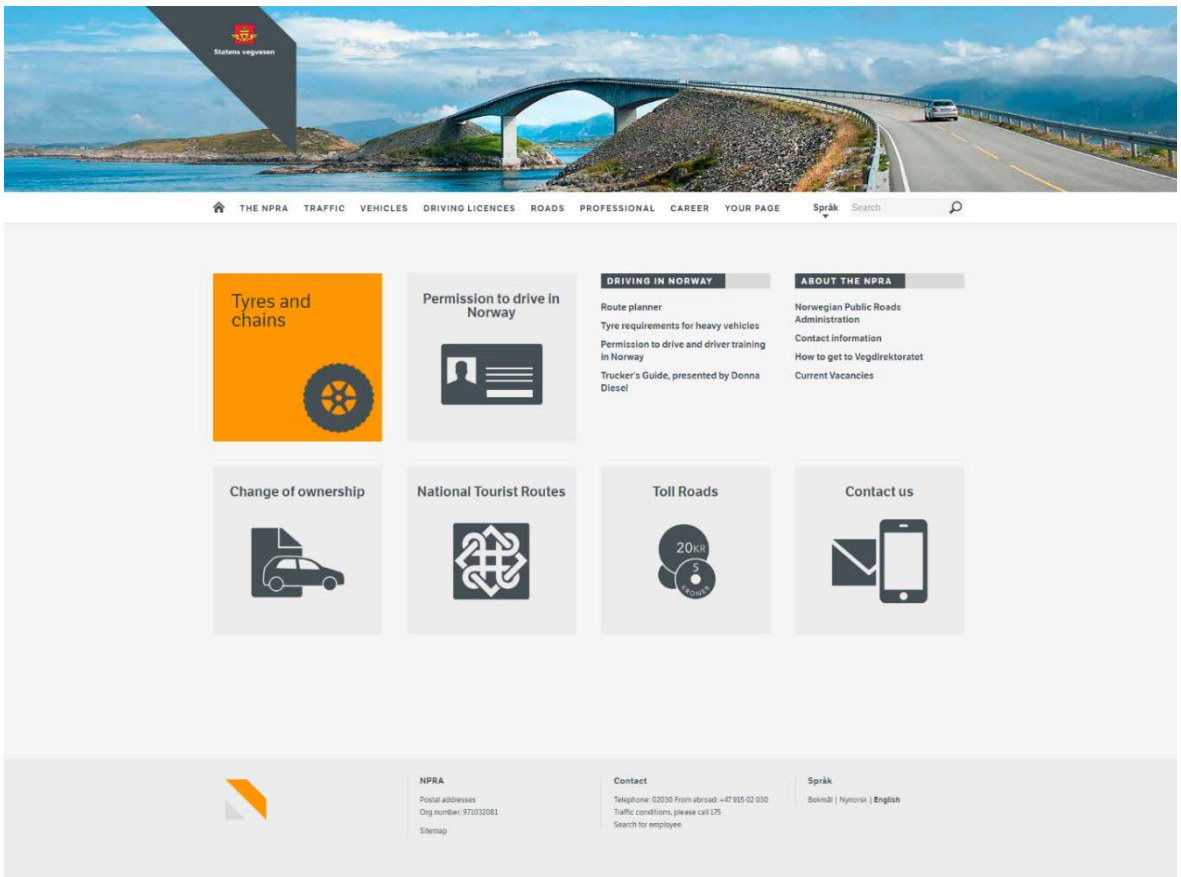


Figure 8-1: User Interface of the web page www.vegvesen.no

8.1 Program Structure

The program structure is displayed in Figure 8-2. The overview presents the pages within the application starting with the *Start page*. The start page gives the user three possibilities;

- go to *Road Object Description* to get road object information.
- go to *Position of Road Object***,** followed up by four pages with *Object Properties*, ending with *Object Property Overview* for road object registration. Additional *Add Object Properties* and *Add Associated Object Properties* to add more relevant properties.
- go to *View Road Object Information Pinned On a Map*, and ending with *Object Property Overview* to view and alter road objects.

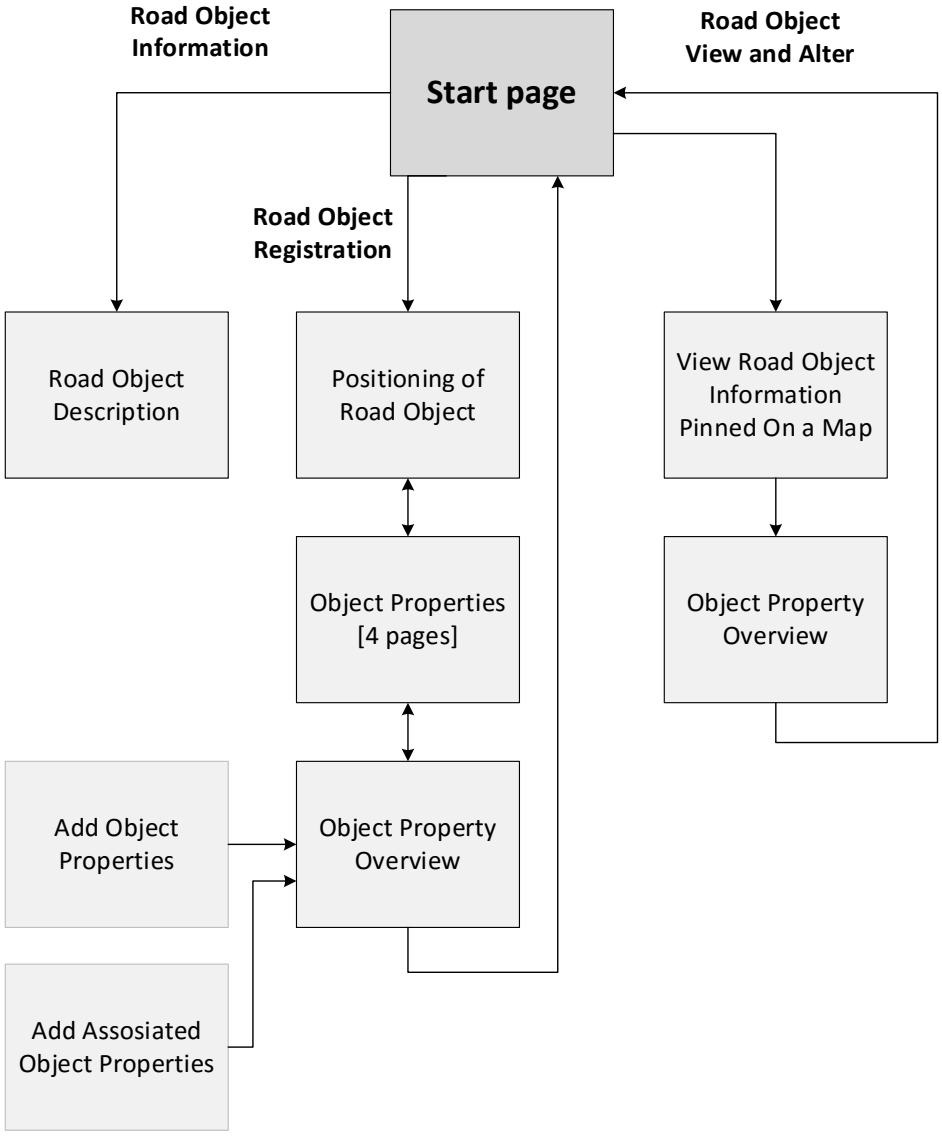


Figure 8-2: Program Structure for the culvert registration tool

8.2 Properties

The application designed in XAML consists of backgrounds, labels and buttons with the colors listed below, which is scales of gray and orange, similar to the NPRA webpage.

#f5f5f5 RGB: 245 245 245

#ececec RGB: 236 236 236

#454f55 RGB: 69 79 85

#ff9600 RGB: 255 150 0

The main font style is Arial. Figure 8-3 displays the main page of the application. The other pages of the application are made in similar style as shown below, using the labels and color styles discussed in this chapter.

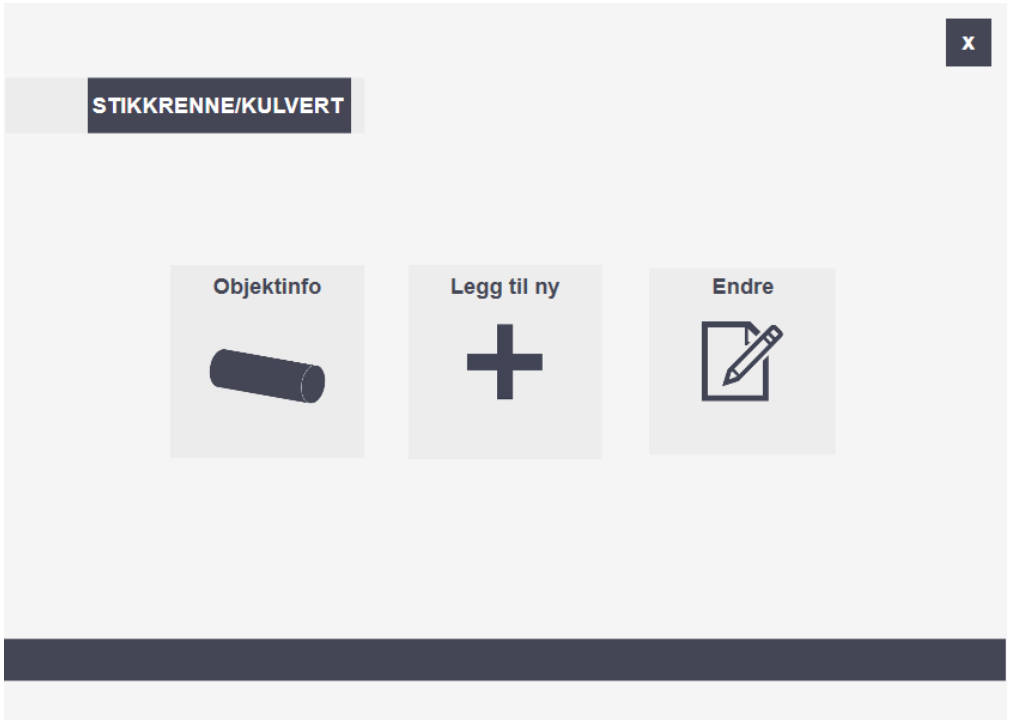


Figure 8-3: User interface of culvert registration tool, start page

8.2.1 Background

The base background of the application looks like Figure 8-4. The background is created with rectangle shapes with the color combination of the three main gray colors presented earlier. The page size is set to height 500 px, and width 670 px, this was done to simulate the application possibilities for a smaller device, such as a tablet.

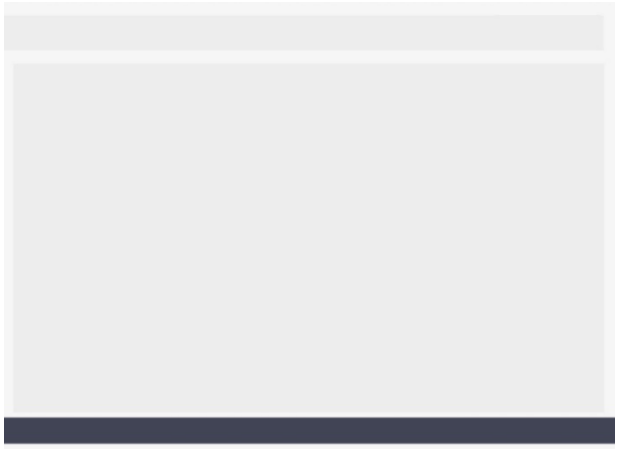


Figure 8-4: Background base for the registration pages

The upper rectangle is the field used for page description and back buttons. The big darker grey rectangle is where the page content is placed.

8.2.2 Buttons

The buttons from the start page shown in Figure 8-5 are designed using Microsoft Visio.

Figure 8-5: Start page buttons

The snippet below shows the XAML sheet for the *object info* button, displaying the size, colors, text content, font details and button background. Similar buttons are made for the *add new culvert* and *view and alter* button.

```
<Button x:Name="btn_ObjektInfo" Content="Objektinfo”  HorizontalAlignment="Left"
Width="130" Height="130" FontFamily="arial" FontWeight="Bold"
Click="button_Click" Foreground="#FF444655" FontSize="14.667">

<Button.Background>
<ImageBrush ImageSource="Objektinfo.JPG"/>
</Button.Background>
</Button>
```

The rest of the buttons within the application consist of the design properties as shown in Figure 8-6, with dark gray background and white font.

Figure 8-6: General button design

The codesnippet for the button properties such as size, background and font details are shown below.

```
<Button x:Name="btnTilbake" Content="Tilbake"  Width="92" Background="#FF444655"
FontWeight="Bold" FontFamily="Arial" Height="37" Click="btnTilbaket_Click"
Foreground="White" FontSize="14.667" BorderThickness="0"/>
```

8.2.3 Icon

The application desktop icon (shown in Figure 8-7) displaying 79 REG for registration of object type 79, which is the object id for culvert. The application icon, follows the same style as the rest of the application.

Figure 8-7: Culvert registration tool ICON

9 Read road data from NRDB

To gather information to use within the application, road data is retrieved from the National Road Data Bank (NRDB). Relevant road data for the culvert registration tool are general information, property types and their possible inputs, existing culverts etc. NRDB REST-API is used to access the relevant road data from the NRDB.

This chapter will introduce the resources available to retrieve the road data, the class used for reading from NRDB in C#, how to retrieve data on different layers within the tree-structure and how to find existing culverts within a specific area.

9.1 Available Resources for NRDB Reading-API

The NRDB API gives access to road data within NRDB. Table 9-1 lists the main resources on the first layer of the NRDB reading-API. Table is from reference [30].

Table 9-1: Available resources for NRDB reading-API

VERB	**URI**	**Description**
GET	/	Gives overview of available resources at root-level
GET	/datakatalog (E: data catalog)	Gives information about the road objects in NRDB and their properties
GET	/vegobjekter (E: road objects)	Access point for road objects in NRDB
GET	/vegnett (E: road network)	Access point for the road network in NRDB
GET	/sok (E: search)	Search interface for finding objects in NRDB
GET	/vegreferanse (E: road reference)	Search interface for road references in NRDB
GET	/omrader (E: area)	Listings of areas in NRDB
GET	/endringer (E:change)	Information about changes in NRDB

NRDB API supports communication using JSON and XML, as shown in Table 9-2. To specify the desired media type, the client has to use “Accept” in the header of the request.

Table 9-2: NRDB REST-API Media types [30]

Type	NRDB REST-API MediaType
json	application/vnd.vegvesen.nvdb-v1+json
xml	application/vnd.vegvesen.nvdb-v1+xml

9.2 Read Road Data Using NRDB Reading-API

The class LesFraApi (translates to Read from API) handles the data requesting to the NRDB reading-API. The class consists of two methods, *RequestString* and *Request*.

RequestString takes relevant URL (Uniform Resource Locator) string as input, as shown in the code snippet below. Creating a variable *request* that uses the abstract class *Webrequest* to make requests to the URI (Uniform Resource Identifier), with the method *CreateHttp* that initializes a new *WebRequest* instance for the URI scheme. [31]

Setting *request.Accept* to application/vnd.vegvesen.nvdb-v1+json.[32] The *resStream* variable is using the method *GetResponse* and *GetResponseStream* to read from the internet request. [33] Variable *rd* creates a new instance of *StreamReader* class to read text, taking *resStream* as an input. [34]

```
static string RequestString(string baseUrl)
        {
            var request = System.Net.WebRequest.CreateHttp(new Uri(baseUrl));
            request.Accept = "application/vnd.vegvesen.nvdb-v1+json";
            var resStream = request.GetResponse().GetResponseStream();
            if (resStream == null)
                return null;
            using (var rd = new StreamReader(resStream))
                return rd.ReadToEnd();
        }
```

The second method *Request*, takes the URL string as the input. The method is using *RequestString* and the class *JsonConvert* with the method *DeserializeObject* to deserialize the JSON object to a .NET object.

```
public static object Request(string baseUrl)
        {
            string s = RequestString(baseUrl);
            if (s == null)
                return null;
            return Newtonsoft.Json.JsonConvert.DeserializeObject(s);
        }
```

9.3 Get First Layer Values from NRDB

The object description page within the program displays the description of the culvert requested from the NRDB API. This description will automatically update when changes are made in NRDB. The code snippet below is from the startup procedure for the description page, where a constant string *stikkrenneUrl* is set to the URL where the description can be found. Culvert description can be located under 'datakatalog/objekttyper/79', which translates to data catalog, object types and 79, which is the object ID for culverts.

```
const string stikkrenneUrl =
"https://www.vegvesen.no/nvdb/api/datakatalog/objekttyper/79";

dynamic result = LesFraApi.Request(stikkrenneUrl);

txtStikkrenneBeskrivelse.Text = Convert.ToString(result.beskrivelse);
```

Defining dynamic *result* by using the *LesFraApi* class described in the previous subchapter, with the *Reques*t method. The document tree structure is shown in Figure 9-1. To get the description (N: beskrivelse) inside the rectangle, *result.beskrivelse* is used.

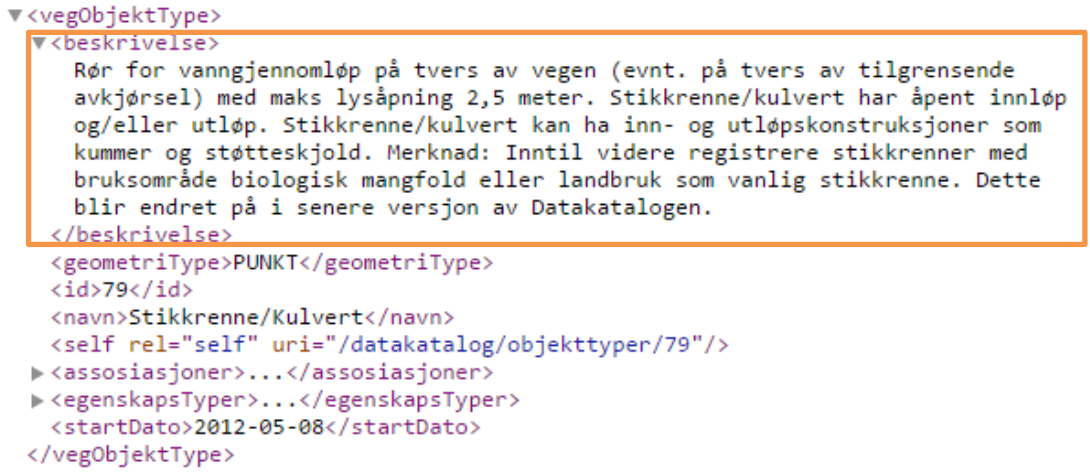


Figure 9-1: First layer in XML document tree, object description of culvert

9.4 Get Property Input Values

In the culvert registration process, the application provides the property input values in comboboxes for the user to choose from. These property inputs are retrieved from NRDB. Figure 9-2 shows the XML tree structure. The properties (N: egenskapstyper) are marked with the big square. The properties have defined entry values, marked with the stapled square.

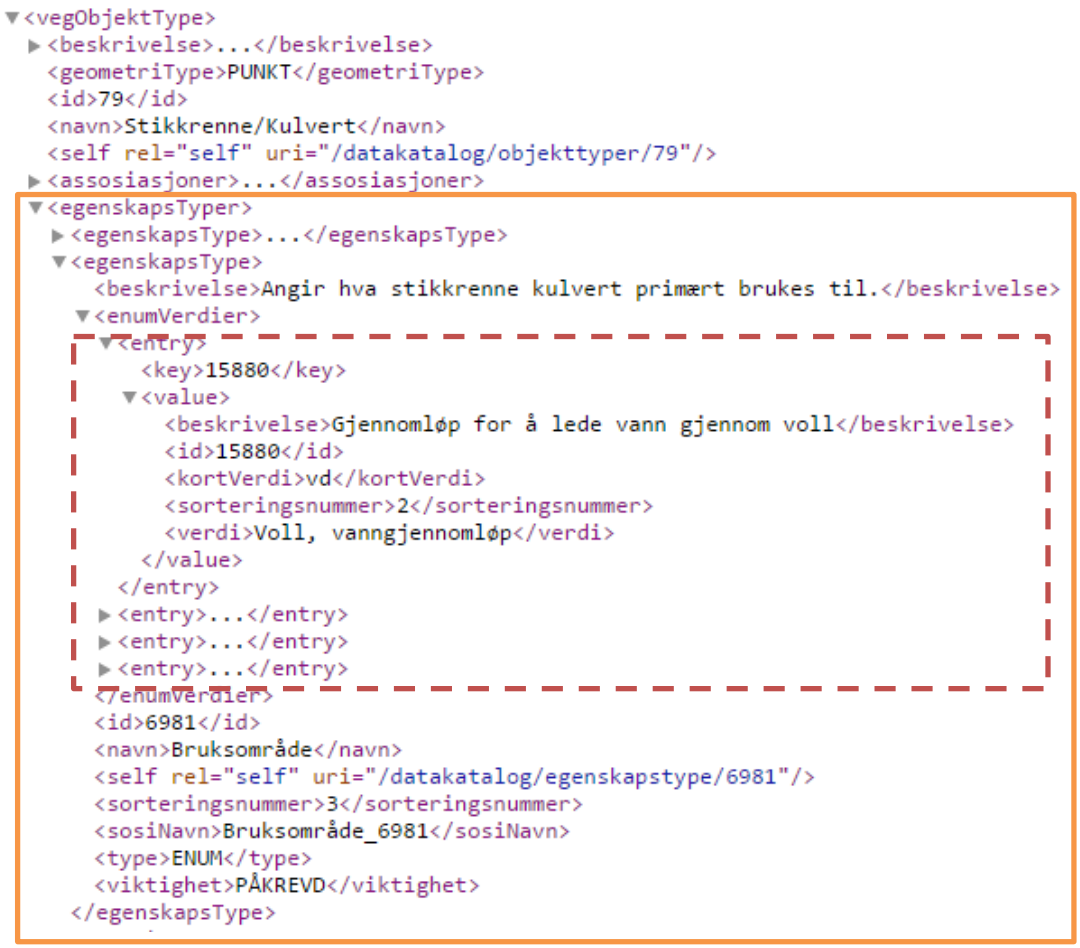


Figure 9-2: Parts of the XML document tree for road object culvert, property entry values

To give an example on how to retrieve these values, the process of catching the input values for *area of use* (N: bruksområde) are explained. The same URL as previous is used, creating a dynamic *result* by using *LesFraApi* to read from NRDB.

The first loop is a *foreach* that loops through the different property types (N: egenskapstyper), using if-statement to find *area of use*. When the right property is found and it consist of input values (N: enumVerdier) another *foreach* is looping through and adding the input values to a combobox.

The properties does not always consist of defined input values, that is why it goes through an if-check to check if the input values are unequal to *null*.

```
const string stikkrenneUrl =
"https://www.vegvesen.no/nvdb/api/datakatalog/objekttyper/79";
            dynamic result = LesFraApi.Request(stikkrenneUrl);

            foreach (var item in result.egenskapsTyper)
            {
                if (Convert.ToString(item.navn) == "Bruksområde")
                {
                    if (item.enumVerdier != null)
                    {
                        foreach (dynamic item in item.enumVerdier)
                        {
                            dynamic entry = item.First;
                            cboBruksområde.Items.Add(entry.verdi);
                        }
                    }
                }
```

This will eventually result in a combobox as shown in Figure 9-3.

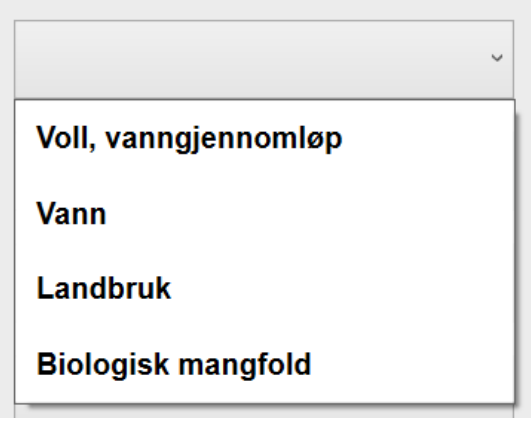


Figure 9-3: Combobox with input values for 'area of use'

9.5 Get Existing Culvert Information from NRDB

The application will ideally give the user the option to view and edit already existing culverts from NRDB. The relevant information is found under the URI Vegobjekter (E: Road objects). When requesting existing culverts, it has to be limited down to a defined area. The selection can be set using geographical coordinates retrieved from a map application within the program.

The application user can interactively move the map around inside of the web browser tool (this will be further discussed in chapter 10), while using methods to send back the corner boundaries. The southwest and northeast coordinates are built into a string that will send a request to the NRDB API to list the culverts in that specific area.

The code snippet below is from *LeafletEndre.html* (further explained in Chapter 10) and displays the function that returns the southwest bound as a string that can be retrieved in C#, a similar function exist for getting the northeast bound.

```
  function GetCurrentBoundSW()
  {
      var cornerSW = map.getBounds().getSouthWest().wrap();
      return cornerSW.toString();
  }
```

The code snippet below is from a method in C# that retrieves the values from *LeafletEndre.html*, by invoking the script for its web browser. A similar method is used for getting southwest bound.

```
Private string GetBoundSouthWest()
 {
    object[] boundSW = { "cornerSW" };
    string boundSouthWest =
Convert.ToString(WebBrowserMapEndre.InvokeScript("getCurrentBoundSW", boundSW));
    return boundSouthWest;
  }
```

When building the request string, a boundingbox with coordinates in either UTM33 or WGS84 is used as shown below:

- UTM33: bbox=[Xmin, Ymin, Xmax, Ymax]&srid=UTM33
- WGS84: bbox=[longitudeMin, latitudeMin, longitudeMx, latidudeMax]&srid=WGS84

As an example: requesting for an area I Porsgrunn with coordinates in WGS84:

```
../nvdb/api/vegobjekter/79/?bbox=9.67863,59.12692,9.69365,59.13243&srid=WGS84
```

Which result in the XML structure shown in Figure 9-4, where there only exist two culverts.

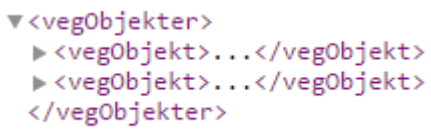


Figure 9-4: XML tree displaying two available road objects within the given bounds

9.6 Get Road Reference

Finding the relevant road reference is done by requesting the NRDB using the geographical coordinates in either WGS84 or UTM33. When providing the coordinate point, the closest road reference is returned. Road reference request belongs to the road reference URL (vegreferanse) as shown below. Table 9-3 shows the resources to get road reference using WGS84 and UTM33 coordinates.

```
https://www.vegvesen.no/nvdb/api/vegreferanse
```

Table 9-3: Available resources for finding road reference based on UTM33-and WGS84 coordinate systems

VERB	URI	Description
GET	/koordinat?ost={østlig-koordinat}&nord={nordlig-koordinat}	Value for east (N: øst) and north (N: nord) in UTM 33-coordinates
GET	/koordinat?lon={longitude}&lat={latitude}	Value for longitude and latitude in WGS84-coordinates

The snippet blow is part of the method *Vegreferanse* in C# where longitude and latitude are used as inputs. The longitude and latitude values are retrieved from map application described in chapter 10. The method builds the URL string with the desired longitude and latitude values and sends request to the NRDB using the reading class *LesFraApi* as explained earlier. The response is returned for further use within the application.

```
public string Vegreferanse(string longitude, string latitude)
{

string UrlVegreferanse = " https://www.vegvesen.no/nvdb/api/vegreferanse/" +
"koordinat?lon=" + longitude + "&lat=" + latitude;

dynamic result = LesFraApi.Request(UrlVegreferanse);
string vegreferanse = Convert.ToString(result.visningsNavn);
return vegreferanse;

}
```

Figure 9-5 shows a snippet from the application where the WGS84-coordinate and the according road reference is given.

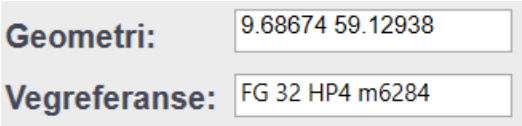


Figure 9-5: Snippet from the application showing geometry and road reference

Creating a URL based on the geometry given above:

```
https://www.vegvesen.no/nvdb/api/vegreferanse/koordinat?lon=9.68674&lat=59.12938
```

This gives the XML tree structure shown in Figure 9-6, where the display name (N: visningsNavn) is shown in the application.

Figure 9-6: XML tree structure for road reference example

10 Map Services

An important part of the application is to pin the location of a culvert to a map and monitor existing culverts based on position. The efficient way to solve these tasks is to use an interactive map.

To be able to display and use a map in the application, a *WebBrowser* tool in C# is used. The *WebBrowser* connects to a local .html file that runs a Leaflet JavaScript to display a map with tiles of Norway provided by Kartverket. All these elements will be discussed in the following subchapters.

For this application, four .html files are in use; Leaflet.html and LeafletMax.html for pinning the location with a marker, one for the regular view and the other for maximum screen display. LeafletEndre.html and LeafletEndreMax.hml for displaying existing culverts on the map, for regular view and maximum view.

10.1 WebBrowser Tool

The *WebBrowser* tool is an inbuilt function in Visual Studio for running web pages. The code snippet below is from XAML displaying some of the properties used for the *WebBrowser* that is going to display the map.

```
<WebBrowser x:Name="WebBrowserMap" Height="317" Margin="23,74,0,0"Width="613"  />
```

When opening the page for the map display, the following C# code is running at startup. This requires the System.IO directory. The *WebBrowserMap* source is directed to the local Leaflet.html file.

```
string curDir = Directory.GetCurrentDirectory();

WebBrowserMap.Source = new Uri(String.Format("file:///{0}/Leaflet.html",
curDir));
```

When running the application, the map will be running inside the WebBrowser.

10.2 Leaflet

Leaflet is one of the most used open-source JavaScript library used for map applications. [35] The following code snippet is from the head of the html file where the connection to leaflet is established.

```
<head> <link rel="stylesheet" href="http://cdn.leafletjs.com/leaflet-
0.7.2/leaflet.css" />
<script src="http://cdn.leafletjs.com/leaflet-0.7.2/leaflet.js"></script>
</head>
```

The map area is defined in the code snippet below, with the width and height varying from 700x500px for the regular window and 1900x900px for the maximum view.

```
<div id="map" style="width:700px; height:500px;"></div>
```

The code snippet below displays the creation of the map. Minimum zoom is set to 0 and maximum zoom to 18. The opening map view will be set to a GPS position in reality, but is now set to a location in Porsgrunn, Norway (Latitude: 59.129672, Longitude: 9.686142). The topology layers are from startkart.no, with layers 'norges_grunnkart', which translates to base map of Norway. Attribution is set to 'Kartverket'.

```
Var map = L.map('map', { minZoom: 0, maxZoom: 18 }).setView([59.129672,
9.686142], 18);
var topolayer = new
L.TileLayer.WMS("http://opencache.statkart.no/gatekeeper/gk/gk.open", {
layers: 'norges_grunnkart',
format: 'image/png',
transparent: false,
version: "1.0",
attribution: "Kartverket"
}).addTo(map);
```

When running the application code, the map will appear as shown in Figure 10-1, where the *WebBrowser* is running the Leaflet.html file.

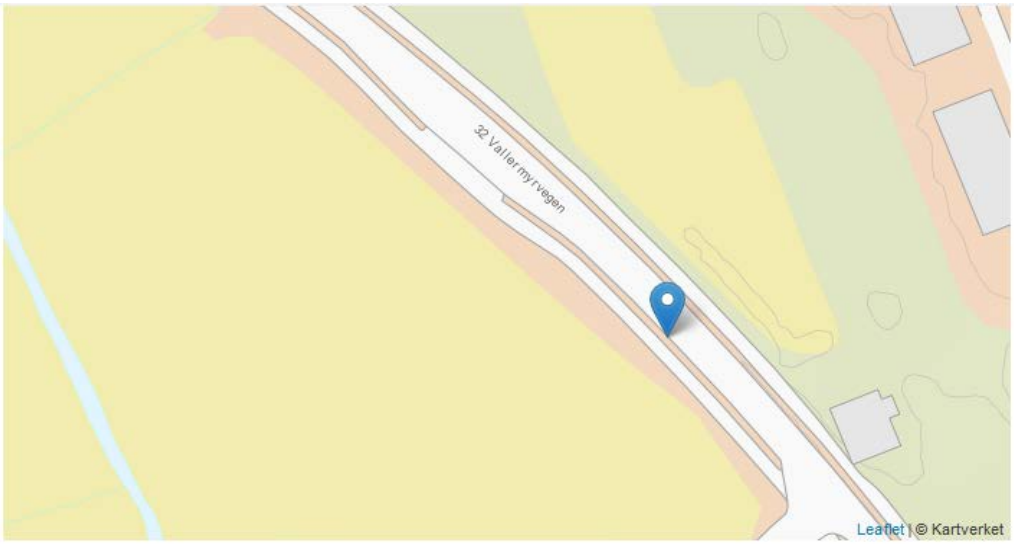


Figure 10-1: Displaying of leaflet map within the application

10.3 Map Tile Provider

Kartverket provides an open-source map service for Norway. It is free for the public to use, as long as Kartverket is credited properly. Within the application, '© Kartverket' must be specified where the map is used. Figure 10-2 shows the crediting in the bottom right corner next to Leaflet. [36]

Figure 10-2: Leaflet and Kartverket references displayed on the map

10.4 Set Location Using Marker

A necessary function within the application is to be able to set the geometry properties of the culverts. The application is ideally retrieving location data from a GNSS receiver and is pinned to the map. However, without access to a GNSS, the location can be manually pinned to the map by using a movable marker. The code snippet below is from Leaflet.html. The marker start position is set to longitude: 59.129380 and latitude: 9.686740 and draggable is set to *true* to be able to move the marker on the map.

The *position* variable is detecting the new position of the marker by using the method *getLatLng*, and using *setLatLng* to give *marker* the new location. The method *panTo*, moves the view to center the marker.

```
marker = new L.marker([59.129380, 9.686740], { draggable: 'true' });
marker.on('dragend', function (event) {
var marker = event.target;
var position = marker.getLatLng();
marker.setLatLng(new L.LatLng(position.lat, position.lng), { draggable: 'true'
});
map.panTo(new L.LatLng(position.lat, position.lng))  });
map.addLayer(marker);
```

The following code snippet shows the function *ondragend*. The function finds the location of the marker by using the *getLatLong* method and returns it as a text string, which the main program in C# is able to collect.

```
function ondragend()
{ var m = marker.getLatLng();
return m.toString(); }
marker.on('dragend', ondragend);
```

The code snippet below is from C#, where the *WebBrowserMaps* script is invoked, which in this case is the Leaflet.html. The arguments for the *InvokeScript* method is the function *ondragend* and the variable *m*.

```
object[] args = { "m" };
string LatLongString = WebBrowserMap.InvokeScript("ondragend", args);
```

10.5 Get Map Bounds

To be able to search NRDB for existing culverts in a specific map area, the current bounds of the map has to be available. The snippet below shows the function *getCurrentBoundSW*, which returns the southwest coordinates as a string.

```
function getCurrentBoundSW()
{ var cornerSW = map.getBounds().getSouthWest().wrap();
return cornerSW.toString(); }
```

A similar function *getCurrentBoundNE*, returns the coordinates for northeast. These values are invoked the same way as explained previously for retrieving longitude and latitude.

10.6 Mark Objects on Map

One of the functions in the application is to mark existing culverts on the map. When clicking the marks, a pop-up with property information about that specific culvert will appear. For each of the items in location under road object, the code snippet below will run. Invoking the web browsers script and for the function *DisplayName*, a new object is created. Sending the ID, latitude, longitude, coordinates in WGS84 and a string with various properties to the html file.

```
WebBrowserMapEndre.InvokeScript("DisplayName", new object[]
                        { id, Lat, Long, geoWgs84, egenskapsString });
```

Below is a snippet from LeafletEndre.html with the function *DisplayName*. The function inputs are id, latitude, longitude, geoUtm33 and property string. The function is creating a circle, pinned to the latitude-and longitude position from the input of the function with a size 10. The color is set to orange, with red fill color. The *bindPopup* is created at the end, with appropriate text and input values.

```
function DisplayName(id, lat, long, geoUtm33, egenskapsString)
{
var circle = L.circle([lat, long], 10, {
color: '#ff9600',
fillColor: '#f03',
fillOpacity: 0.5
}).addTo(map);
circle.bindPopup("id: " + id + "<br />" + "Geometri: " + geoUtm33 + "<br />"
+egenskapsString);
}
```

Figure 10-3 shows the running application, with one culvert displayed with a red circle on the map. The information pop-up box appears when the user clicks the red circle.

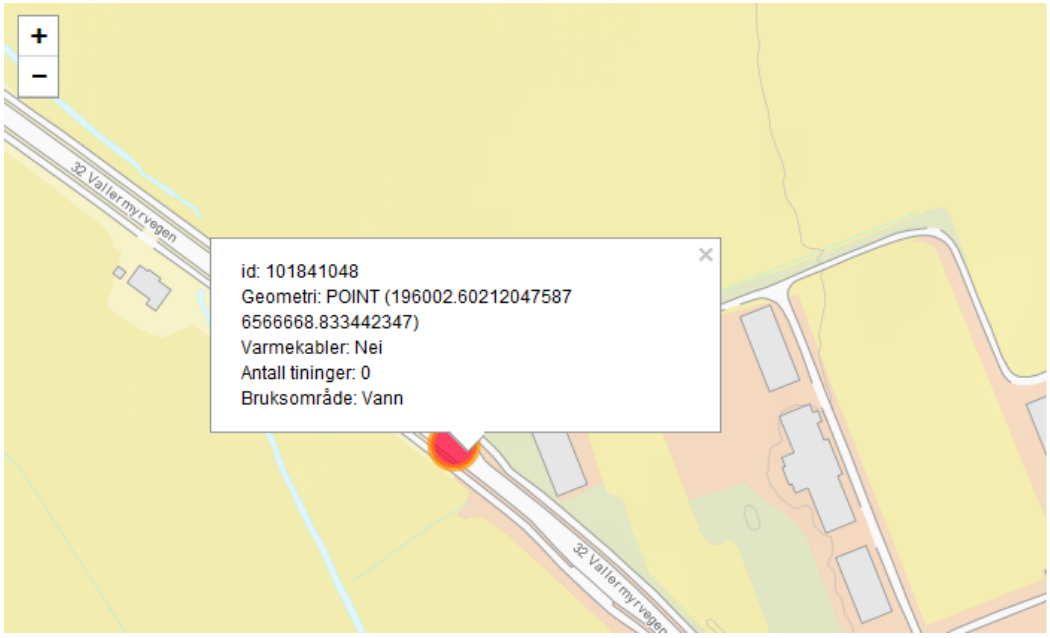


Figure 10-3: The map with a pop-up containing information regarding the specific culvert

The displaying of culverts has a limitation of 100 culverts from the NRDB. Figure 10-4 displays numerous culverts scattered within the map boundaries.

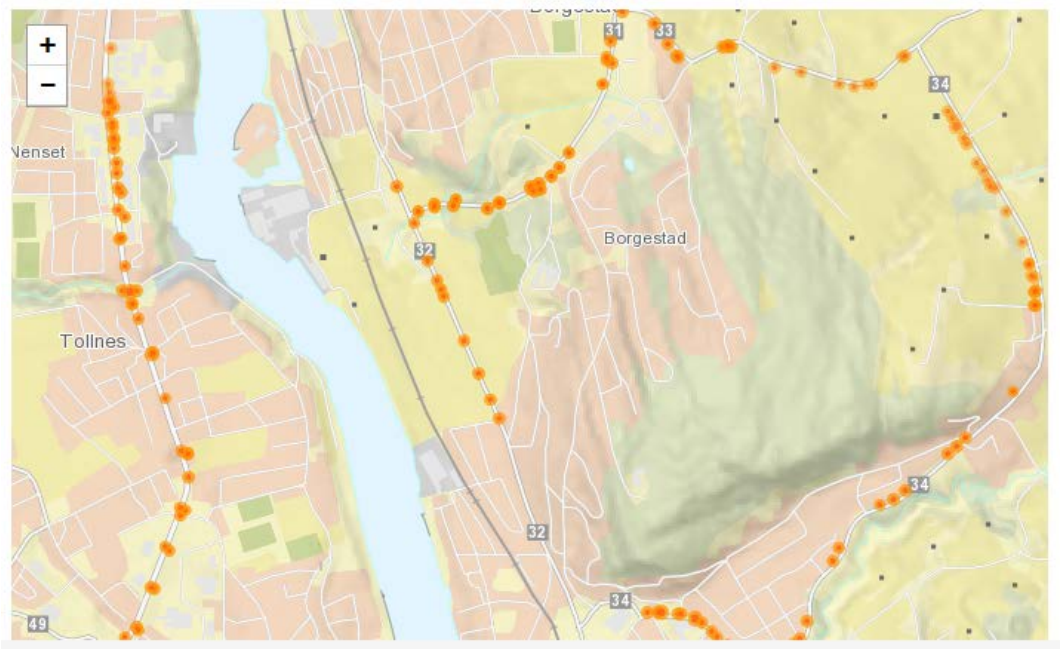


Figure 10-4: The map displaying numerous culverts

11 Write road object information to NRDB

At the end phase of the registration process, the object specifications such as geometry and property values are saved to NRDB. In reality, this is done similarly as the reading process, but using a writing-API and sending requests to store the information. The ability to write information to the NRDB is not possible without authorisation. However, NPRA released a developer tool for the writing API (March 2016), making it possible to establish communication and saving new objects.

This chapter will present how to set up and connect to the developer tool, the resources available in the writing-API as well as the process of saving a new object to the NRDB writing-API developer tool. Most of the references within this chapter is from inside the developer tool. When the developer tool is installed, these references can be reached.

11.1 The Procedure of Running the Developer Tool

The following chapter will present an overview on how to set up the writing-API developer tool using Windows and how to establish communication by using C#.

First step is installing Docker from https://docs.docker.com/engine/installation/, when the installation is complete run Docker Quickstart terminal. The terminal uses a few second to boot up, when it is ready, run the NRDB writing-API by using the command below:

```
docker run –d –p 8080:8080 nvdbapnevegdata/nvdb-skriveapi
```

Figure 11-1 displays the Docker Quickstart terminal after a successful command run.

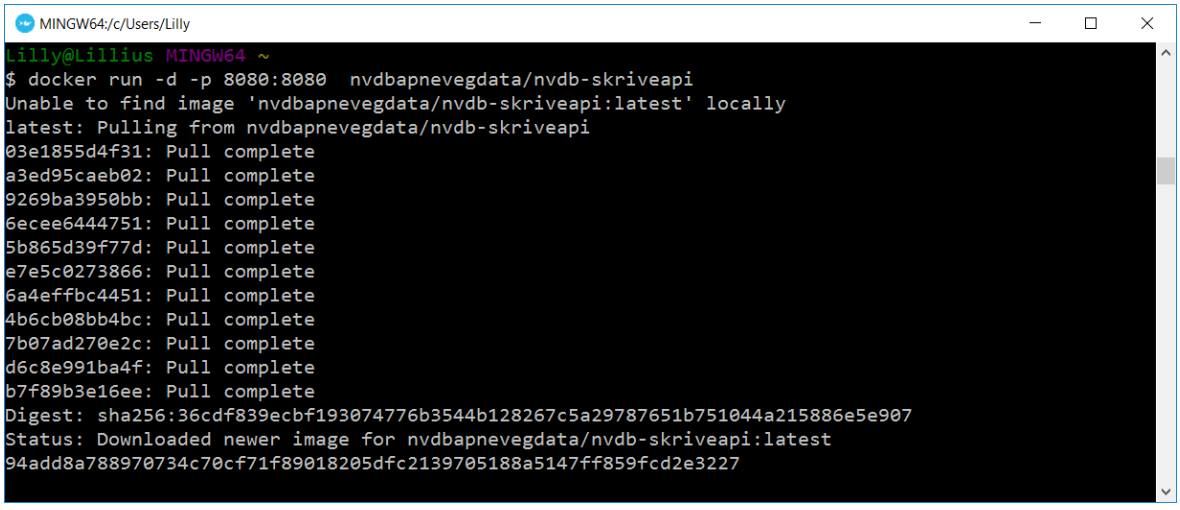


Figure 11-1: Docker Quickstart terminal after running the command for writing-API

The writing-API is available on http://{docker-machine IP}:8080. Open Command Prompt to request the docker-machine IP address by using the command:

```
docker-machine ip
```

Figure 11-2 displays Command Prompt, with the given IP address of 192.168.99.100.

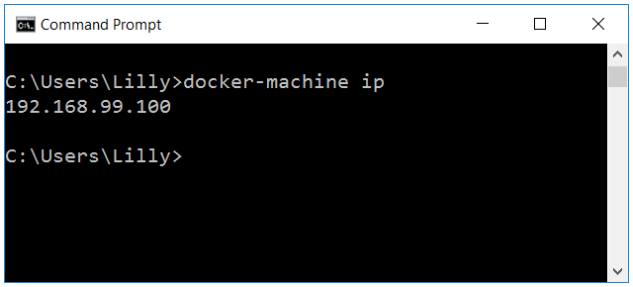


Figure 11-2: Command Prompt, showing the docker-machine IP

The writing API is now available to access in the web browser at http://192.168.99.100:8080/. The web content that will appear is displayed in Figure 11-3, where there is an option to enter NVDB-API (NRDB), to log in and log out.

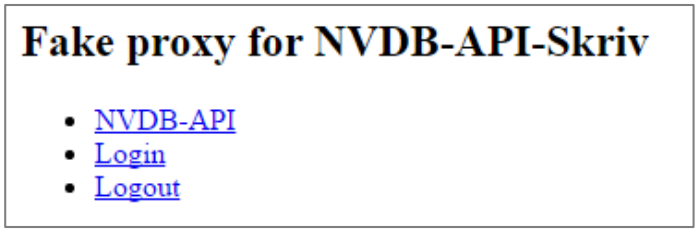


Figure 11-3: Web content of http://192.168.99.100:8080/

When first accessing the webpage in the web browser, it is required to log in. The login page is shown in Figure 11-4. It gives you the opportunity to log in with any given username. There is a possibility to log in as a task system user under the username ‘joruns’, or as a system administrator as ‘extscs’. The username ‘root00’ gives you the same rights as an internal user, ‘exroot00’ gives you the rights of an external user and ‘Tjeroot00’, of a ‘faceless’ task system.

Figure 11-4: Login page for the NRDB writing-API developer tool

When the login procedure is complete, it opens the page shown in Figure 11-5, where there is a possibility to go to the generator page, which provides numerous sets of test cases for writing to the API. The second link is the control panel that documents all the change sets and their statutes, as well as requests made to the API, locks, data rights and setups. The last link is to the documentation on how to use NVDB API SKRIV (NRDB writing-API). It presents all the available resources on how to communicate with the NRDB writing-API, with explanation and related examples.

Figure 11-5: NVDB API Inn

11.2 Available Resources for the Writing-API

Table 11-1 gives an overview of the available API resources for the writing-API. Table is from reference [37].

Table 11-1: Resources for the writing-API

VERB	URI	Description
POST	/nvdb/apiskriv/v2/endringssett/validator	Validate a change set
POST	/nvdb/apiskriv/v2/endringssett	Register a change set
GET	/nvdb/apiskriv/v2/endringssett	An overview of changes
POST	/nvdb/apiskriv/v2/endringssett/{endringssettId}/start	Start processing of a change set
POST	/nvdb/apiskriv/v2/endringssett/{endringssettId}/kanseller	Cancel processing of a change set
POST	/nvdb/apiskriv/v2/endringssett/{endringssettId}/restart	Restart processing of a change set
GET	/nvdb/apiskriv/v2/endringssett/{endringssettId}/fremdrift	Progress of a change set
GET	/nvdb/apiskriv/v2/endringssett/{endringssettId}/status	Status of a change set
GET	/nvdb/apiskriv/v2/endringssett/{endringssettId}	Content of a change set
POST	/nvdb/apiskriv/v2/konverter/sosiGeoJson	Convert SOSI-GeoJSON to change set format
POST	/nvdb/apiskriv/v2/konverter/sosi	Convert SOSI-NVDB to change set format
POST	/nvdb/apiskriv/v2/binaer	Upload binary data
POST	/nvdb/apiskriv/v2/binaer	Upload multipart/form-data
GET	/nvdb/apiskriv/v2/binear/{ressirsId}	Download binary data

In the culvert registration application, the focus will be on validating a change, register the change and then start the procedure of writing to the database.

11.3 Validate

Validation of a change set is necessary to inspect the set for possible errors or shortcomings. The validation process will give either the progress status ‘UTFØRT’ or ‘AVVIST’. ‘UTFØRT’ translates to conducted, which tells the user that the set was conducted and accepted; with this response the user can continue the procedure of adding the change set to the database. ‘AVVIST’ translates to denied, and tells the user that the validation of the change set got rejected, and is therefore not able to continue with the procedure. The validation is a POST request to the API, with the path:

```
../nvdb/apiskriv/v2/endringssett/validator
```

Content type is described in the request header, which describes what media type to be used for the request. The media types available are listed in Table 11-2.

Table 11-2: Media type options [37]

Type	**MediaType**
JSON	application/json
XML	application/xml

An example of a structure of a change set in XML is shown below. The change set gives the property type ‘6980’, which is the *name* property for the culvert, a value. Egenskap (E: property), is where all the different properties related to the culvert will be listed.

```
<endringssett xmlns="http://nvdb.vegvesen.no/apiskriv/domain/v2"
effektDato="2016-05-02" datakatalogversjon="2.01">
  <registrer>
    <vegObjekter>
      <vegObjekt typeId="79" tempId="-1">
        <egenskaper>
          <egenskap typeId="6980">
            <verdi>StikkrenneNavn</verdi>
          </egenskap>
         </egenskaper>
        <lokasjon>
          <punkt lenkeId="1125767" posisjon="0.3"/>
        </lokasjon>
      </vegObjekt>
    </vegObjekter>
  </registrer>
</endringssett>
```

11.4 Register a Change Set

When the change set is validated and accepted, the next step is to register the change set. This is done by using a POST request to the path:

```
../nvdb/apiskriv/v2/endringssett
```

There is a need to specify the content-type in the header of the request. The change set is registered in the writing-API with a given unique ID and is waiting for the start procedure. Figure 11-6 shows the generator within the web browser displaying the change set with ID and progress status ‘not started’ (IKKE_STARTET).

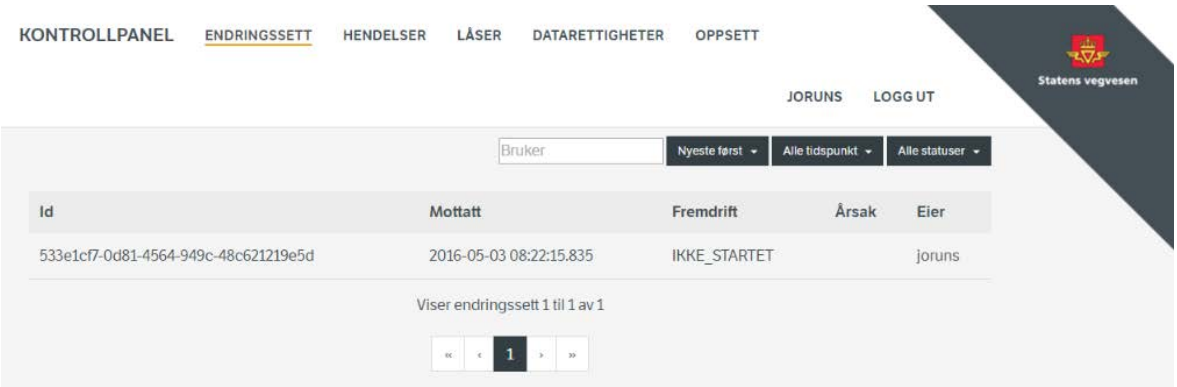


Figure 11-6: Displaying a change set with the progress status ‘not started’

11.5 Start Processing the Change Set

When the change has gone through validation and registration, the final part is to start the processing of the change set. The start procedure is a POST request to the path:

```
../nvdb/apiskriv/v2/endringsett/{change set ID}/start
```

Which in this example would result in:

```
../nvdb/apiskriv/v2/endringsett/533e1cf7-0d81-4564-949c-48c621219e5d/start
```

Figure 11-7 shows the generator page, where the change set has turned green and the progress status is ‘completed’ (UTFØRT).

Figure 11-7: Displaying a change set with progress status 'Completed'

When clicking the change set, it gives a more detailed overview with ID, received time and date, the owner, progress status and the complete change set with potential warnings or errors. There is also a list of detailed events of the communication from beginning to end of adding the change set to the database.

11.6 Other Actions

Other actions available are ‘cancel’ (N: Kanseller) which has to be used before the start processing has begun, to cancel the registration. ‘Restart’ makes processing of a change set that is stuck on hold (VENTER), to start over again. ‘Progress’ (N: Fremdrift) is a way for the user to request a simple overview of the change set progress, which is used for monitoring. ‘Status’ on the other hand gives a more detailed progress status of the change set. There are also possibilities to check out a specific change set or even list change sets from a specific time period. All these actions are well documented in the NVDB API SKRIV documentation.

11.7 Connecting to NRDB Writer-API C#

When requesting to the developer writer-API, the URL request has to be directed to the docker. Which makes the path look something like shown below:

```
http://192.168.99.100:8080/nvdb/apiskriv/v2/...
```

11.7.1 Login

For each request to the writing-API from the application, the page requires another login. It is therefore necessary to create a cookie to store the login information.

In the main C# program, the chosen username for the login is set and uploaded to the root login page within the writing-API, http://192.168.99.100:8080/login.

```
CookieAwareWebClient client = new CookieAwareWebClient();
var values = new NameValueCollection {{ "user-id", "joruns" }};
client.UploadValues("http://192.168.99.100:8080/login", values);
```

Below is a snippet of the request header, which will be explained in more details later. *CookieContainer* is a proptery that retrieves or sets the relevant cookies for the http request. [38]

```
var request = WebRequest.CreateHttp(new Uri(URL));
request.CookieContainer = client.CookieContainer;
```

11.7.2 Building a XML String for Registration of Object

To create a change set for registration, there is a need to build a XML message for the POST request to the writing-API. The example below will go through the creation of an XML string for creating an object with chosen properties. The process would be similar for a JSON string, the only difference is by using JSON format.

The beginning of the XML-string starts as follow:

```
string xml = @"<endringssett xmlns=""http://nvdb.vegvesen.no/apiskriv/domain/v2""
effektDato=""2016-05-05"" datakatalogversjon=""2.01"">
<registrer>
<vegObjekter>
<vegObjekt typeId=""79"" tempId=""-1"">
<egenskaper>";
```

The registration procedure is specified by using <register>. Next is specifying the type of road object, which is 79 for culverts. The end of the start-string opens for properties (N: egenskaper) to take place.

The snippet below shows the procedure of adding properties.

```
if (GlobaleVariabler.Bruksområde != null)
{xml += @"<egenskap typeId=""6981""><verdi>" + GlobaleVariabler.Bruksområde +
"</verdi></egenskap>";}

if (GlobaleVariabler.Navn != null)
{xml += @"<egenskap typeId=""6980""><verdi>" + GlobaleVariabler.Navn+
"</verdi></egenskap>";}
```

When the properties have been set in the application, the procedure shown above will go through all the culvert properties to check if it holds a value. If this is the case, the according string with property ID (typeId) and relevant value (verdi) is added to the start string. The string builds on as the program goes through all of the properties. When the checking of all properties is completed, the end string as shown below is added.

```
xml += @" </egenskaper>
<lokasjon>
<punkt lenkeId=""1125766"" posisjon=""0.3""/>
</lokasjon>
</vegObjekt>
</vegObjekter>
</registrer>
</endringssett>";
```

The change set in XML-string format is now complete and can be used to post to the API-write for validation.

11.7.3 Altering Existing Object

The procedure for altering an existing object string is similar to the procedure described above for registration. The main difference is in the start for the XML as shown below. Switch <register> with <korriger>, meaning correction and use NRDB ID (nvdbId) input to write to a specific object.

```
<endringssett xmlns="http://nvdb.vegvesen.no/apiskriv/domain/v2"
effektDato="2016-05-05" datakatalogversjon="2.01">
<korriger>
<vegObjekter>
<vegObjekt typeId="79" nvdbId = "551800127" versjon="1">
<egenskaper>
```

The procedure for adding/altering properties and the XML ending will be the same as explained for the registration procedure.

This procedure is readymade in the application, but not set to use, as the developer tool for the writing-API is not able to do the physical changes, which would only be possible if writing to the actual writing API.

11.7.4 Validation of Change Set

The code snippet below is from the most important part of the method *Validering*, where the validation procedure of the change set is executed. The XML-string for the change set is retrieved from the method *buildString*, which is returning the built object string. The URL for the validation part is shown below:

```
http://192.168.99.100:8080/nvdb/apiskriv/v2/endringsett/validator
```

The *WebRequest* is made to this URL. The validator procedure uses *POST* method and the content type application/xml or application/json. These need to be specified to make the request, as well as setting the *ContentType*, *Method*, *CoookieContainer* and the length of the *requestBytes*. The class *Stream*, and *requestStream.Write* is used to transfer the change set.

```
private void Validering()
{
byte[] requestBytes = Encoding.UTF8.GetBytes(buildString());
string validatorUrl =
"http://192.168.99.100:8080/nvdb/apiskriv/v2/endringssett/validator";
var request = WebRequest.CreateHttp(new Uri(validatorUrl));

request.ContentType = "application/xml";
request.Method = "POST";
request.CookieContainer = client.CookieContainer;
request.ContentLength = requestBytes.Length;

Stream requestStream = request.GetRequestStream();
requestStream.Write(requestBytes, 0, requestBytes.Length);
requestStream.Close();

WebResponse response = request.GetResponse();
requestStream = response.GetResponseStream();
StreamReader reader = new StreamReader(requestStream);
string responseFromServer = reader.ReadToEnd();
```

The validation process will only check if the change set is in accepted format in relation to the NRDB and the data catalog. The change set will not be stored within the NRDB.

Either the validation response will give the user ‘AVVIST’, for rejected or ‘UTFØRT’, for conducted, giving the user the option to add or change the change set.

11.7.5 Change Set

The only difference in the procedure for endringsett (E: change set) is the URL, as shown below.

```
string EndringssettrUrl =
"http://192.168.99.100:8080/nvdb/apiskriv/v2/endringssett";

var request = WebRequest.CreateHttp(new Uri(EndringssettrUrl));
```

This time the change set will be stored and set to waiting status. It is now possible to look at the change set from within the developer tool for the writing-API. The change set is now given a unique ID, which the program retrieves in the response from the request. The ID is used in the next step, which is the start procedure.

11.7.6 Start

At this point, the change set is ready and waiting to be started. The code snippet below with the *Start* method shows the start procedure. It uses the change set ID retrieved at the former request and implements it in the start URL:

```
http://192.168.99.100:8080/nvdb/apiskriv/v2/endringsett/{endringsett ID} /start
```

Using similar classes and methods as explained earlier, this time not writing, only requesting for the already stored change set to begin the process of saving to the NRDB.

```
private void Start(string ID)
{
string UrlStart = "http://192.168.99.100:8080/nvdb/apiskriv/v2/endringssett/" +
ID + "/start";

var request = System.Net.WebRequest.CreateHttp(new Uri(UrlStart));
request.ContentType = "application/xml";
request.Method = "POST";
request.CookieContainer = client.CookieContainer;

Stream requestStream = request.GetRequestStream();
requestStream.Close();

WebResponse response = request.GetResponse();
requestStream = response.GetResponseStream();
}
```

The process of writing the current road object, in this case, a culvert, is now fully completed. The property variables within the culvert registration tool are set to *null* and the user is taken to the start page, where it’s possible to register, supervise or alter new objects.

12 Culvert Registration Tool

The main idea when dealing with an on-the-go field application is that it should be simple to operate. Meaning that the main and most important components should be in focus, as probably more detailed registration work will take place later on a more complex program. Even if that is the case, the application still provides enough options to be a fully functional program (for its purpose) and not just a start tool. To be more specific, the most important functionality that should be in focus for this application, are the ability to add new culverts and to do monitoring and altering of existing culverts. An additional place to read about general culvert information is also handy. Figure 12-1 shows a simple overview of the program structure.

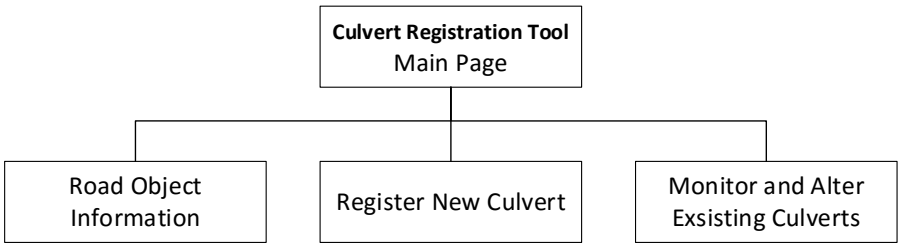


Figure 12-1: Basic program structure

Figure *12-2* shows the start-up page of the application featuring these exact possibilities. The first button is for object information (N: Objektinfo), the second button for adding new culvert (N: Legg til ny) and the last for monitoring and altering existing culverts (N: Endre).

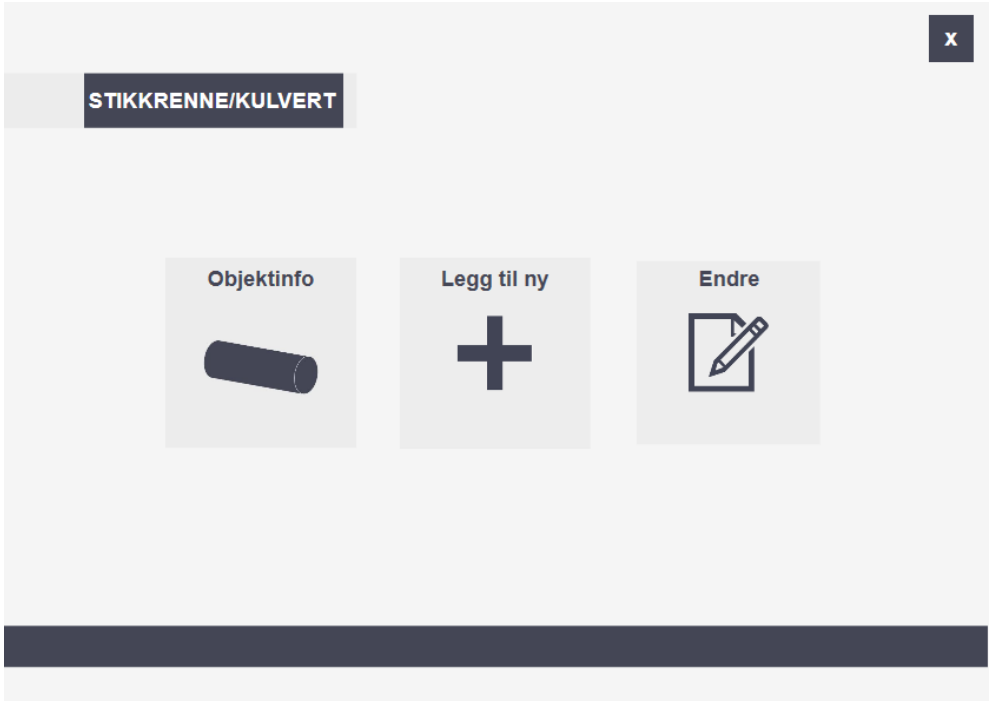


Figure 12-2: Start-up page for the culvert registration tool

12.1 Road Object Information

The road object information page is shown in Figure 12-3. This page provides the user with the road object description, which is retrieved from NPRA regarding culverts.

Figure 12-3: Road object information page

12.2 Add New Culvert

The most natural way to start the registration while on the road is to pin the location of the object, as shown in Figure 12-4. The registration process starts by setting the *start* and *end* location. To get the precise location data, a GNSS should be used. However, within the application, getting location data is carried out by pinning a marker to the desired location on the interactive map and receiving corresponding coordinates to use in the registration tool. This method is possible to use for some road objects, but as culverts require position accuracy of 20 cm, this would not be sustainable.

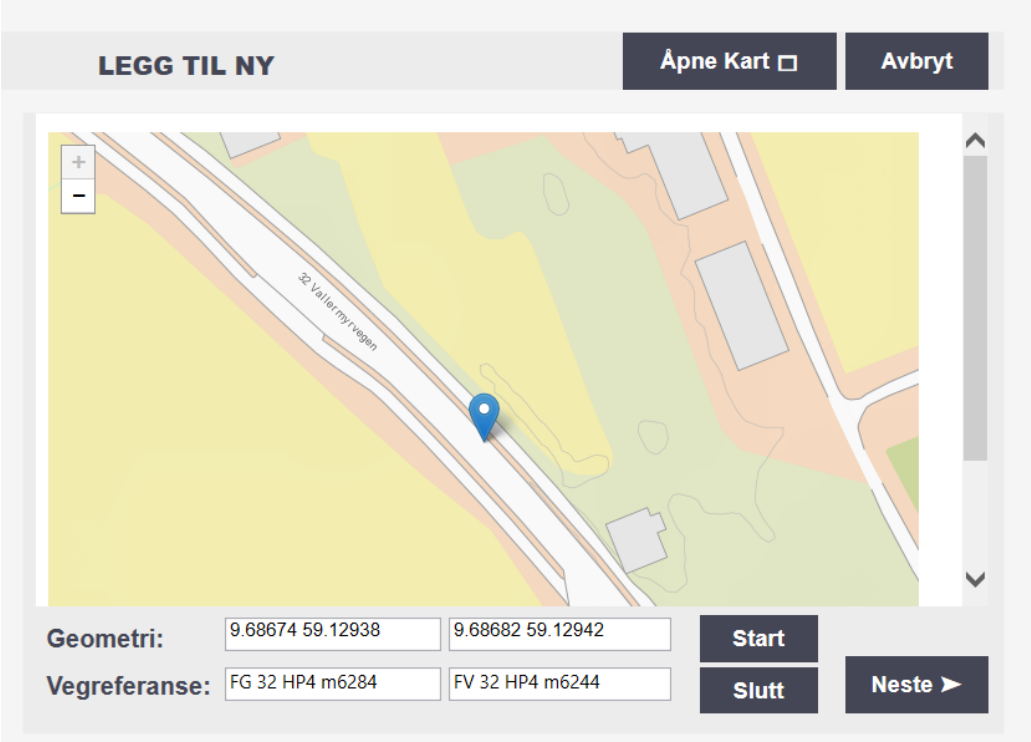


Figure 12-4: The first registration page for geographical positioning of the culvert

There is an option to maximise the map to fill the entire screen, making the map handling easier, by clicking the button *Open Map* (N: Åpne Kart). When the map is in full screen, shown in Figure 12-5, there is an additional option to view culverts that are registered in the NRDB map area. This screen view is only possible/relevant depending on the device the application is running on.

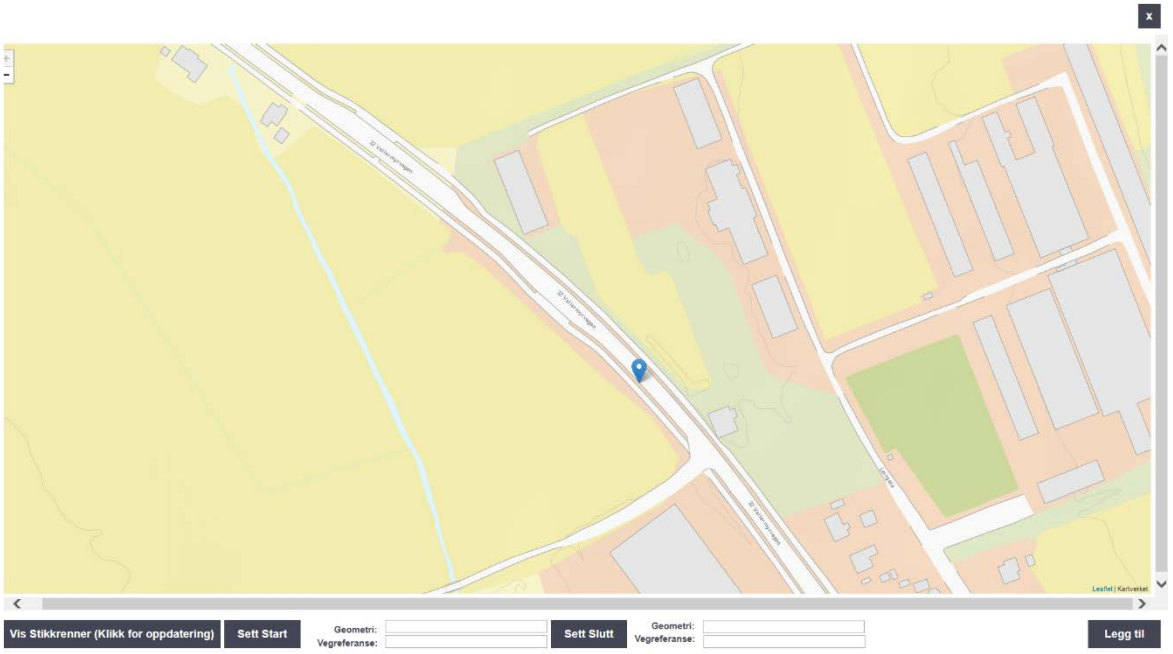


Figure 12-5: Full screen geographical positioning of the culvert

Every road object in the database usually consist of a comprehensive list of properties regarding the specific object. The properties have different labels of importance, as listed below (list from product specification in Appendix B):

- Absolutely required (need to exist to be able to save)
- Required (strongly recommended, but still able to save without)
- Conditional (needed if relevant for the object, but still able to save without)
- Optional (no value required)
- Optional Special Information (no value required)
- Expires (properties on the way out of NRDB)

For the road object culvert, there is no 'absolutely required' properties, meaning there is no absolute need for any of the properties to have a value to be able to register. Which leads us to the next level, 'required', which is the most relevant properties to add in the registration, the property registration process starts with these. The next focus area is the conditional properties.

The focus when creating the application was not to make a general application for road administration, even if this would be ideal to have an application that could adapt to different road objects. Because the focus was on the culverts specifically, each of the pages in the registration process are customised to be as user friendly as possible by adapting the properties regarding culverts.

This leaves us with five pages to fill in properties of choice as shown in Figure 12-6. The first page concerning the road geometry, property 1 and 2 for required properties and property 3 and 4 for conditional properties, ending off with the overview. The user can move freely back and forth between the property pages in case of the need to alter. When reaching the overview, more properties can be added if needed.

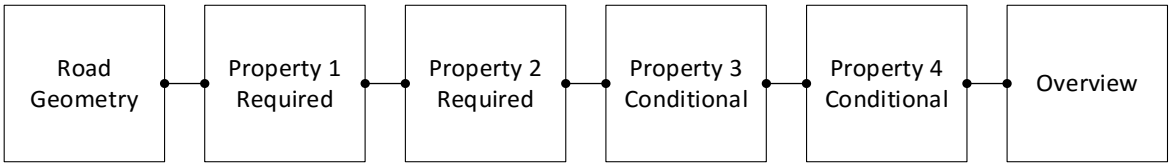


Figure 12-6: Overview of the registration process within the application

A major topic in the application planning was the idea of a simple tablet friendly application, handled with a regular touch screen, meaning bigger fonts, larger buttons and fewer distractions. In general, a more simple and interactive interface. Even if using several pages require more clicks than what an endless list of properties would, it is still

easier to distinguish between the relevant properties and to sort out what is needed in the specific categories.

The first page displays four of the ‘required’ properties; area of use, year of construction, type of material and shape, shown in Figure 12-7. These properties can be filled in using comboboxes that lists possible inputs defined by the NRDB. Year of construction only requires a numeric value, but holds a list for the last four years, to help speed up the process. There are conditional properties associated with the choice of culvert shape; these will appear as pop-up boxes that will require properties such as diameter, width and height.

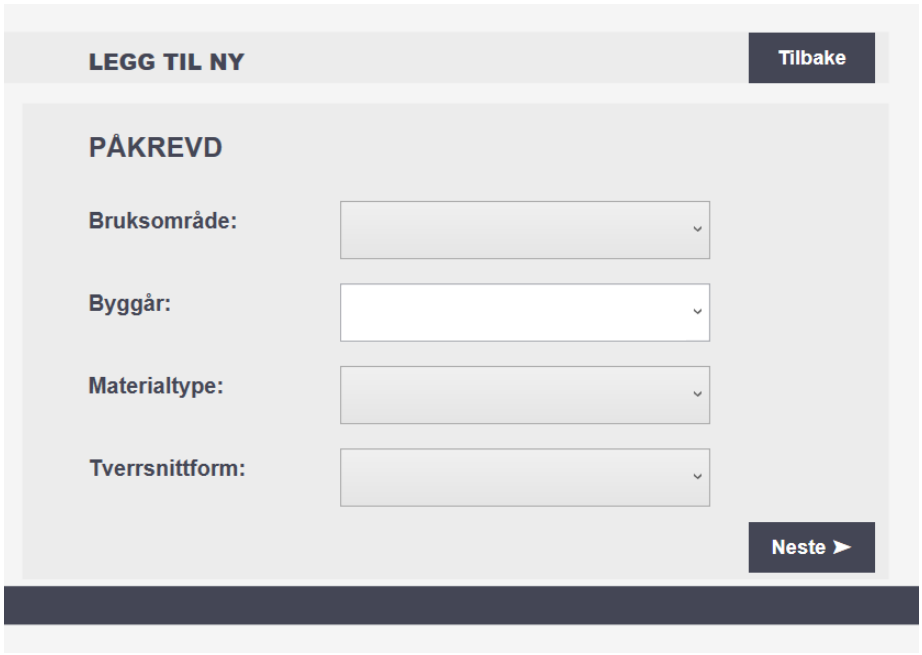


Figure 12-7: Property page 1, required properties

The pop-up with conditional properties regarding rectangular shape is shown in Figure 12-8, where width and height inside of the culvert has to be set with the unit mm. Similar pop-ups will appear for circular and ellipse shape, with their conditional properties.

Figure 12-8: Pop-up box with conditional properties regarding culvert shapes

The second page is the last of ‘required’ properties, type of inlet and outlet, shown in Figure 12-9. These can simply be set using comboboxes holding defined allowed inputs, before moving on to conditional properties. If the registration process is already complete at this point, the user can go directly to the overview page to submit, by clicking ‘Ferdig’.

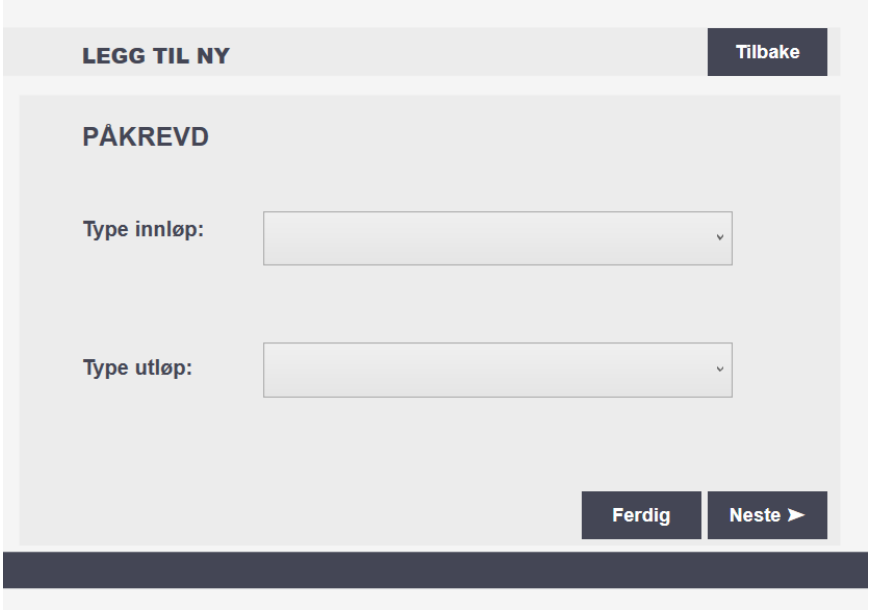


Figure 12-9: Property page 2, required properties

The third page is for conditional properties having possible inputs as ‘Yes’ or ‘No’, as shown in Figure 12-10. Buttons are used to increase the number of clicks. The conditional properties are; connected to closed drainage, passageway for river/stream, having inlet grade and lastly having heating cables.

Figure 12-10: Property page 3, conditional properties

The forth page holds the last of conditional properties, numbers of thawing, owner and maintainer, shown in Figure 12-11. The two last pages are only relevant if it differ from the ordinary. The relevant property values are set using comboboxes.

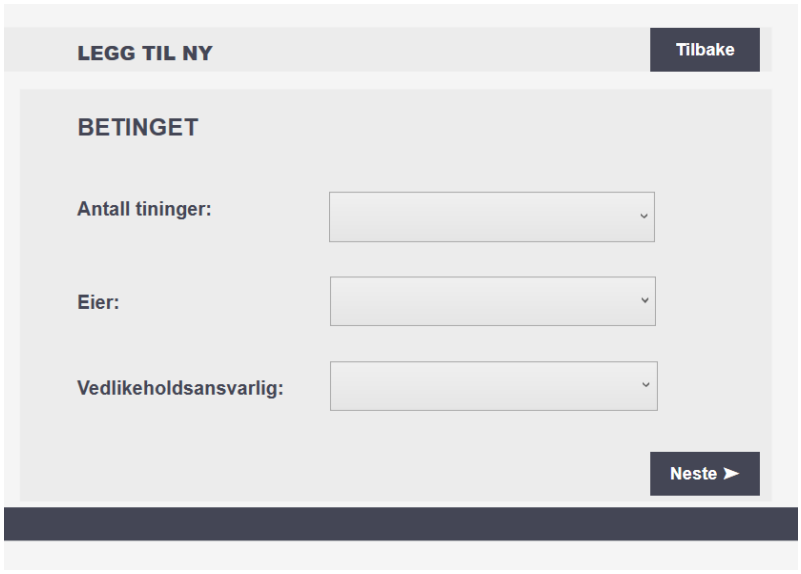


Figure 12-11: Property page 4, conditional properties

The last stop is the main overview, as shown in Figure 12-12. From this part on, the main purpose of the application is usually fulfilled and the information for an on-the-go registration is satisfied, whereas the user can continue by adding the road object to the database. However, there are additional possibilities; choose a property in the list and remove it, add more properties from other categories and add associated objects of the culvert.

Figure 12-12: The Overview page, displaying the chosen property values

When there is a need to add more properties for the registration, the ‘Legg til Egenskaper’ button is clicked, and the page shown in Figure 12-13 appears. The list presents all

possible properties for culverts, retrieved from NRDB. When choosing a property from the list, the corresponding property description will appear underneath ‘beskrivelse’, and possible input values will appear in the combobox under ‘verdi’. If the property doesn’t consist of a preset with possible inputs, the combobox works as a text field, where the value can manually be written in. The buttons underneath the property list correct the list to show:

- ALLE (E: ALL) – lists all defines properties from NRDB
- P – Påkrevd (E: Required) – lists all properties of importance required
- B – Betinget (E: Conditional) – lists all properties of the importance conditional
- O – Opsjonell (E: Optional) – lists all properties of the importance optional
- S – Opsjonell Spesialinformasjon (E: Optional Special Information)

The buttons display the capital letter of each category, even if these are not explained within the application. The users of the program are most likely familiar with the categories and are able to adapt easily. When the desired value of the property is set, the ‘Legg Til’ button is returning the user to the overview where the new property added now is displayed.

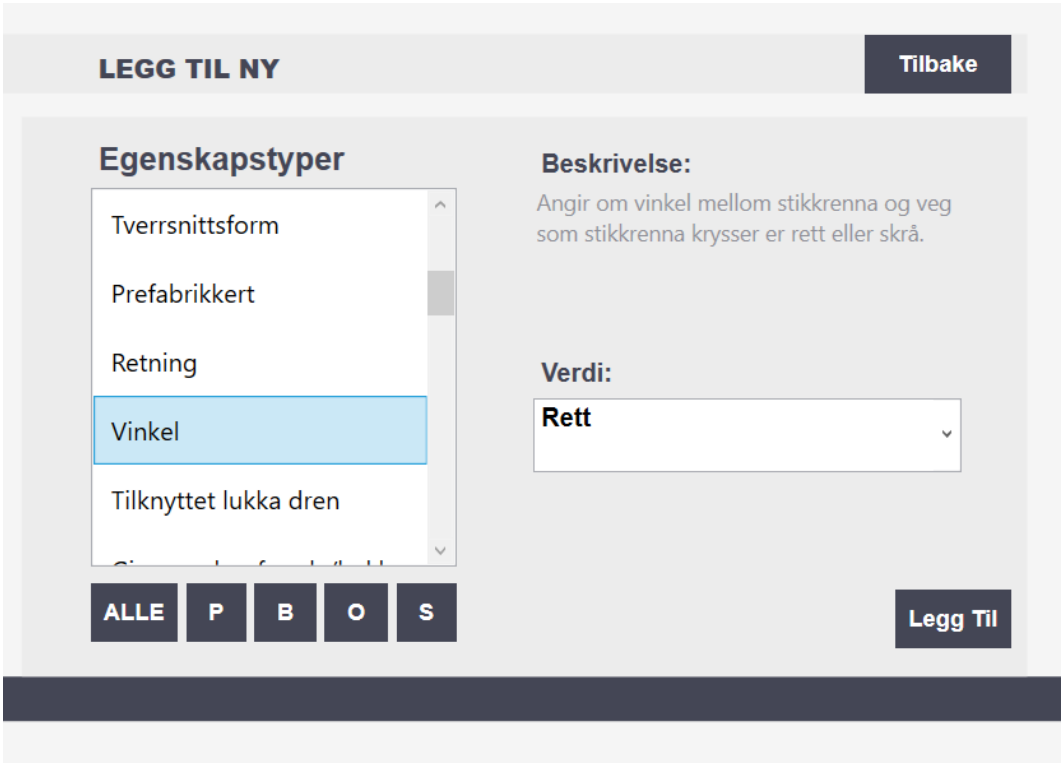


Figure 12-13: The page for adding more culvert properties

In addition to adding new properties of the culverts, the user is able to add associated objects with properties. The add associated objects page is shown in Figure 12-14. The listbox on the left-hand side, lists all the associated objects of the culvert. When the desired associated object is chosen, the object description will appear below the listbox.

Corresponding properties is shown in the drop down menu ‘Tilhørende egenskaper’ (E: Associated properties) and possible inputs of the chosen property is shown in the combobox below. If there is no pre-set input values, the combobox works as a text field where the user can write in the desired value. The ‘+’ button is adding the properties without returning to the overview page. This fastens up the process if there is more properties of one associated object that is needed, or more associated objects needed. When everything is added, the ‘Legg til’ button returns the user to the overview.

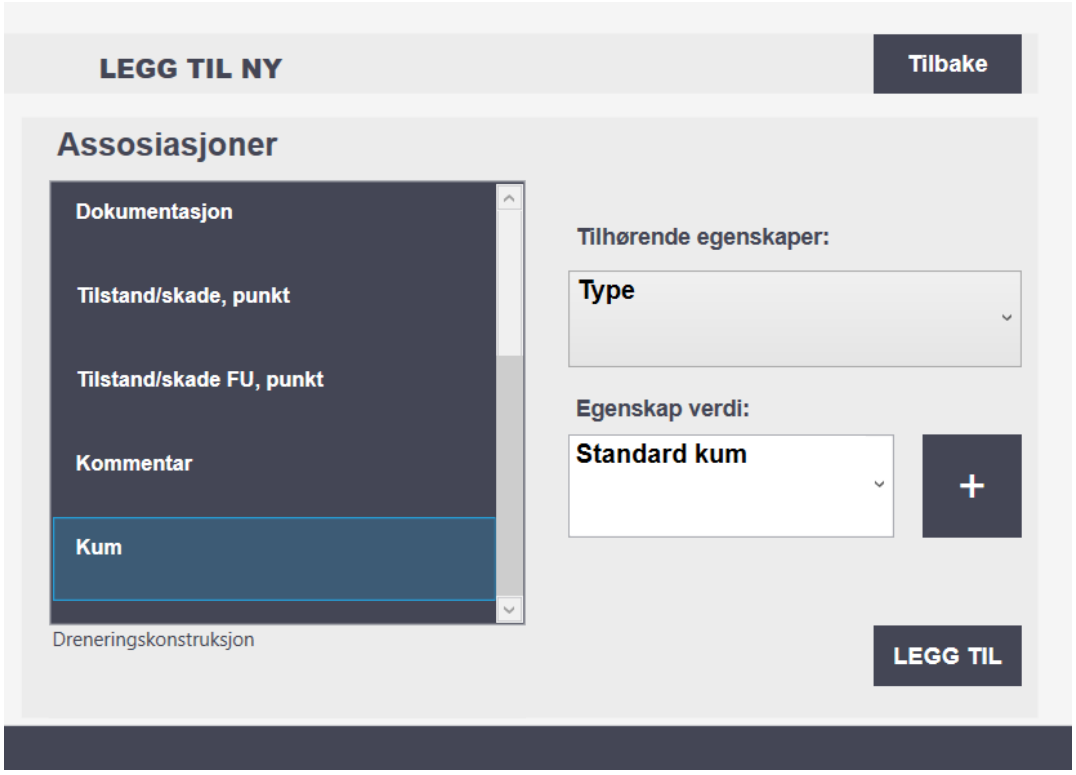


Figure 12-14: The page for adding associated road objects with property values

In the previous figure, the associated object ‘Kum’ (E: Manhole) was chosen and the property ‘Type’ was given the value ‘Standard kum’. These details will appear in the overview as shown in Figure 12-15. The added associated object will have its own object point with its associated properties displaying underneath.

Figure 12-15: The overview after associated property is added

12.3 Monitor and Alter Existing Culverts

The last option of the culvert registration tool is to monitor and alter existing culverts. The monitor and alter page is shown in Figure 12-16. This page have a similar option to maximize the map as discussed earlier. Within the program, the map is set to a standard opening view, which is a location in Porsgrunn, Telemark. For future work, when a GNSS is connected, the start view can be set directly to its location, using the geographical coordinates of the GNSS as a center for the map.

When clicking the ‘Vis Stikkrenner’ (E: Show Culverts), the culverts within the map boundaries are pinned on the map with red/orange circles.

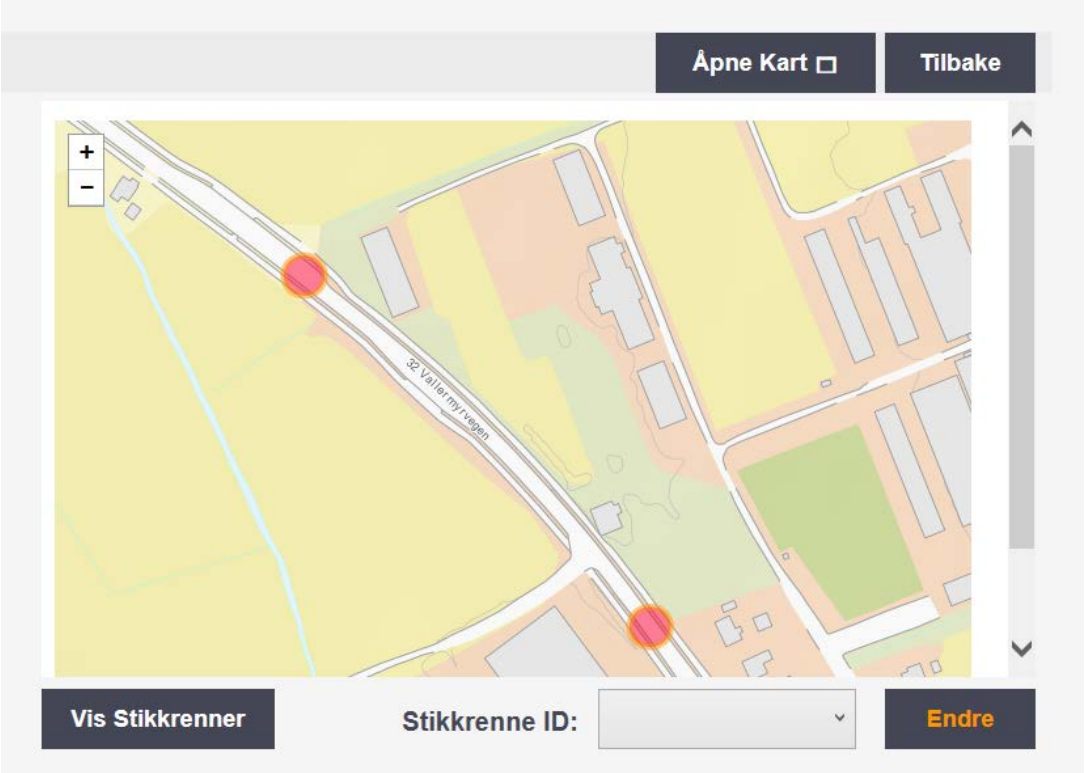


Figure 12-16: Monitor and alter page, displaying two culverts

Figure 12-17 shows the map zoomed out and displaying numerous culverts. The application is able to display maximum 100 culverts at the time, because of limitations from NRDB. However, the application is most likely used when driving around to monitor the culverts, using a much closer view, as shown earlier.

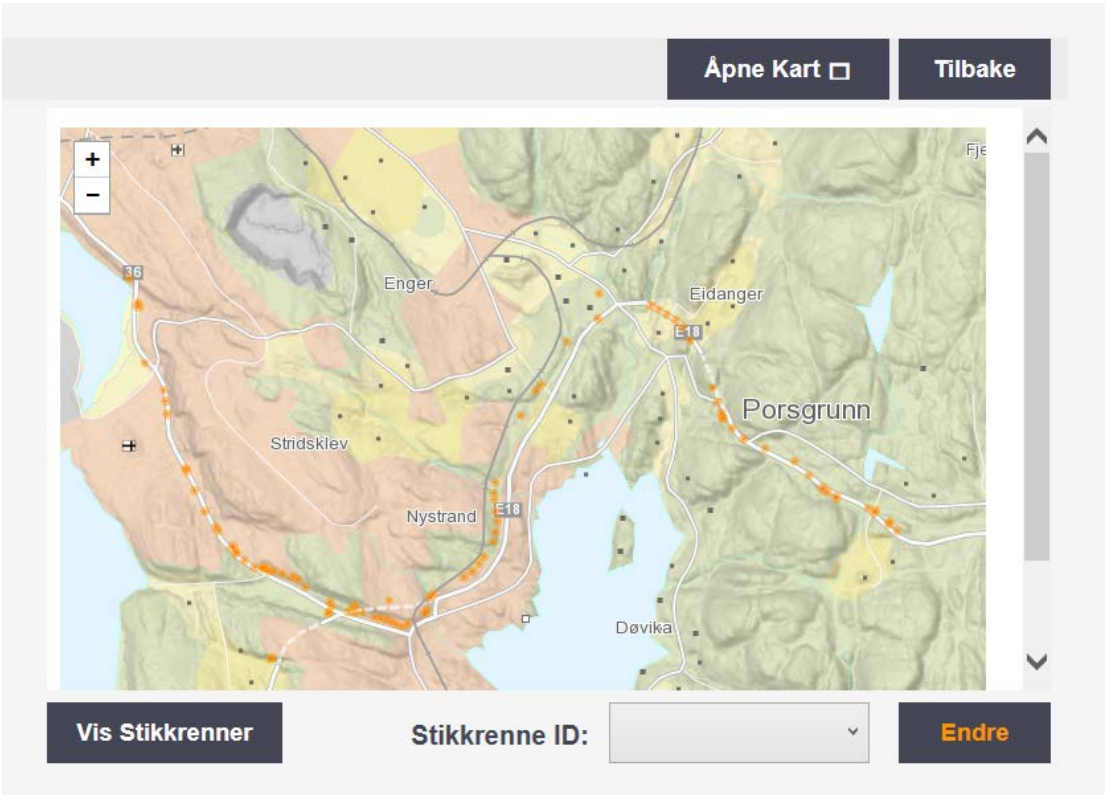


Figure 12-17: Monitor and alter page, displaying several culverts

When monitoring the culverts, its ID and properties is shown in a pop-up window as shown in Figure 12-18. The ID of the culverts within the map boundary is shown in the combobox, where the desired culvert can be chosen by its ID number and checked out closer in the overview page by clicking ‘Endre’ (E: Alter).

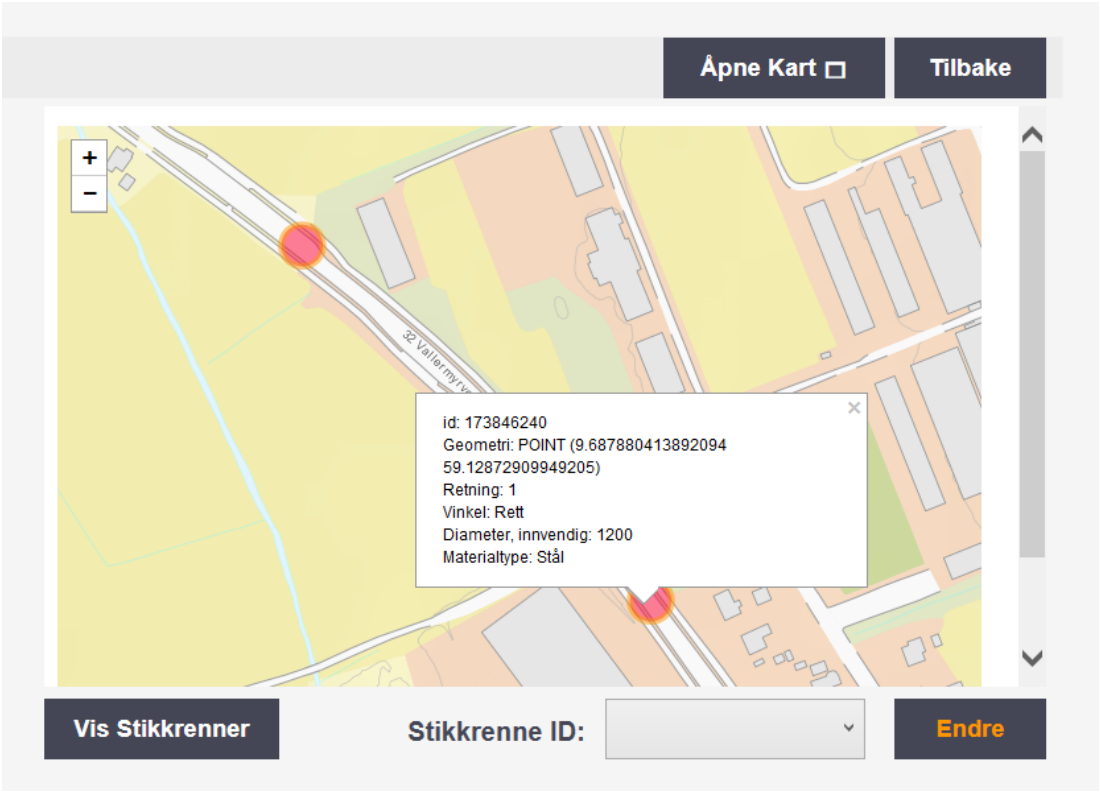


Figure 12-18: Monitor and alter page, displaying a pop-up for culvert

Figure 12-19 shows the overview page with the properties as shown in the previous figure. This page makes it easier to monitor and if needed; add more properties, remove properties or add associated objects. If no changes are needed, the button ‘Avbryt’ (E: Cancel) is clicked.

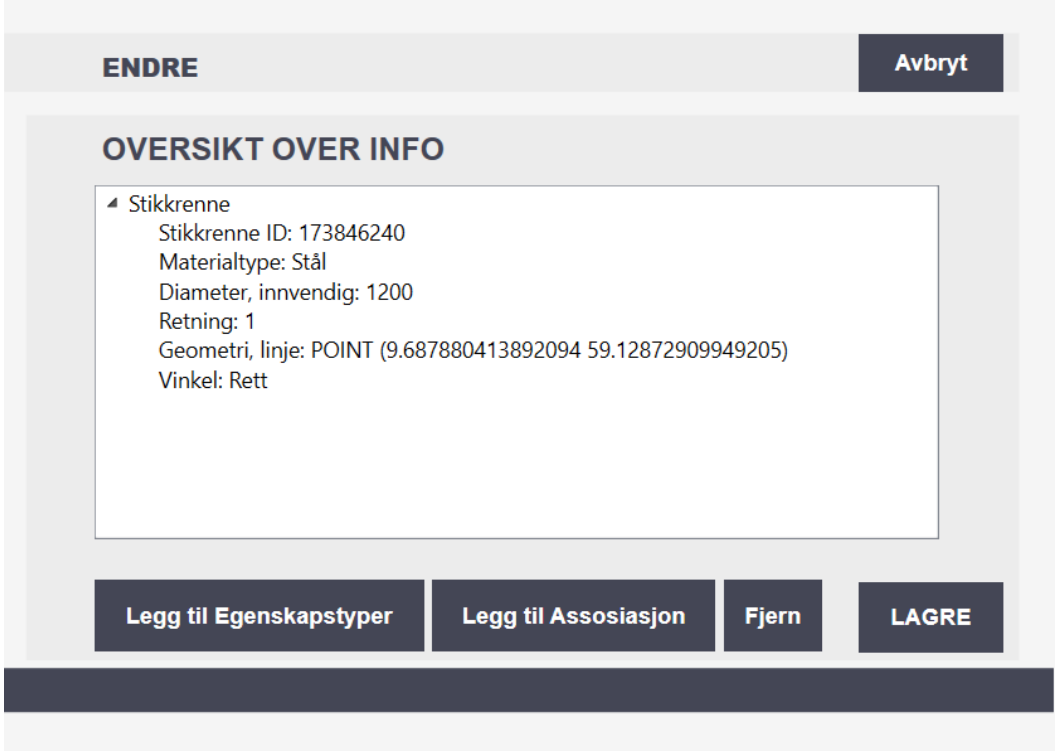


Figure 12-19: Overview of existing culvert values

For future work, when the application is able to write to the NRDB, a functionality within the program should correct the saving procedure; to use alter, instead of a register procedure, based on the type of page leading to the overview (as for now, it’s only changing the header and buttons).

A complete systematic user manual is available in Appendix C. The user manual takes the user through every step of the registration process and explains every functionality within the application.

13 Testing and Deployment

At the end of the software development lifecycle is the testing and finally the deployment of the application. The software has continuously been tested internally during the entire development of the application. However, a more complex testing procedure is required before officially launching the application. The testing is incomplete as the developer tool for the writing-API was launched close to the project end.

This chapter will present basic testing theory, levels and methods. Ending with the deployment/installation of the application, culvert registration tool.

13.1 Software Testing

Testing is a crucial part of software development, it is necessary for discovering malfunction and bugs within the software. Testing is a continues process throughout the development process of the application. There are different types of errors to take into consideration; requirement, design, code, documentation and bad-fix errors. [24]

The testing is executed by the programmer during the development and in the end phase. In addition, there are usually assigned workers who will test the system. Testers are usually someone with technical background, their main responsibility is to write test cases and make sure they are executed. To ensure that the customers' requirements are fulfilled, the customers usually take part in the testing process, to ensure that they are satisfied with the product. [24]

13.2 Testing Levels

There are several testing levels; unit, regression, integration, system and acceptance testing, overview shown in Figure 13-1.

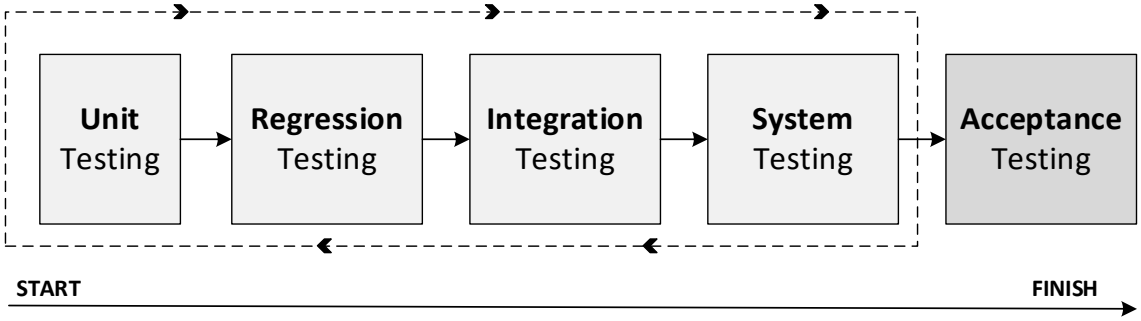


Figure 13-1: Testing levels, from start to finish [24]

- **Unit Testing:** Isolate parts of the software and test it independently from the whole application, making sure all the functionalities within the program is working standalone.
- **Regression Testing:** Tests to make sure that everything is working as it should after a change is applied to the system.
- **Integration Testing:** Combining the parts from the unit testing, to check if the parts are working together properly. Integration testing can also be testing interactions with other systems.
- **System Testing:** Tests the complete system, checking if the software meets the requirements of the system.
- **Acceptance Testing:** Customer test, to make sure the software meets the requirements of the user/customer. [24]

13.3 Testing Methods

There is a wide range of testing methods such as; unit testing, functional/non-functional testing, integration testing, regression testing, white-box/black-box testing, code review and acceptance testing (FAT/SAT) to mention some.

13.3.1 White-Box and Black-Box Testing

Black-Box testing, is testing executed by someone without programming background that mainly tests the application functionalities. No prior knowledge of the program is needed for Black-Box testing. The method mainly focuses on the input and output of the program and not the inner program structure.

White-box testing on the other hand, takes the internal structure into consideration. This type of testing has to be executed by someone with knowledge of general programming and the development of the specific software that is tested. [24]

13.3.2 FAT/SAT Testing

FAT, which stands for Factory Acceptance Testing is executed at the software company within a test environment. SAT, which stands for Site Acceptance Testing is testing executed within the production environment by the customers themselves.

When the acceptance test is executed and the customer is satisfied with the results, the software is ready to be launched. [24]

13.4 Testing of Culvert Registration Tool

The application has continually been tested throughout the development. Starting with unit testing for each of the different functionalities within the program, integration testing for different parts working together, and system testing for the program as a whole. The 'customer' NRPA had the opportunity to do one test round.

Using both white-box and black-box testing, by involving colleagues with a certain base of programming knowledge and the of the software content and involving people without any knowledge of the content of the software, only its requirements and expected system behaviour. A test case document example with test description and filled in comments are available in Appendix D.

The test document involves test cases based on functionality checks, such as:

- Are the buttons leading to the right places and/or preforming desired functionality
- Are the maps loading properly and are the internal map functions responding and working as planned
- Are the desired values needed, retrieved from the NRDB properly
- Are the text fields adapted to its input values
- Are the different parts and functionalities working together as they should

The application was tested on different Windows versions, to make sure the application would be operable on older various as well as newer. The application was tested on Windows 10, 8 and 7. Besides some small interface differences, the application functionalities worked satisfying. When installing the application on older computers with old .NET versions, it was directed to the download of the needed .NET to be able to run the application.

The testing was done without the NRDB writing-tool, because of its late availability, all required functions when writing to the developer tool is not working optimally. The testing for this part is in the early stages, and requires a much more complex testing procedure to ensure the right registration formats to the NRDB.

13.5 Deployment/Installation

The target framework of the application is .NET framework 4.5.2, which is the version the .NET framework target for the application. However, the application have supported runtime version of .Net framework 4.0.

The installation folder, consist of three main parts, as shown in Figure 13-2. The application files, the main application and the setup file.

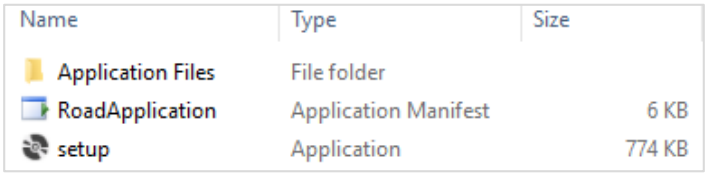


Figure 13-2: Installation folder content

When double clicking the setup file, the application install window as shown Figure 13-3 in will appear. Click the install button to start the installation.

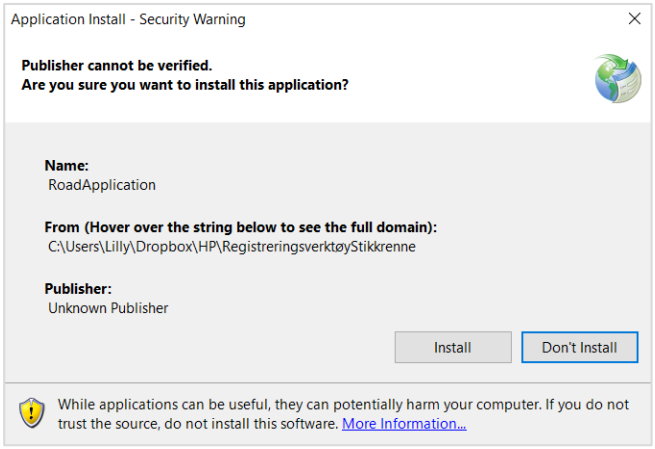


Figure 13-3: Application Install – Security Warning

The application is starting to install as shown in Figure 13-4.

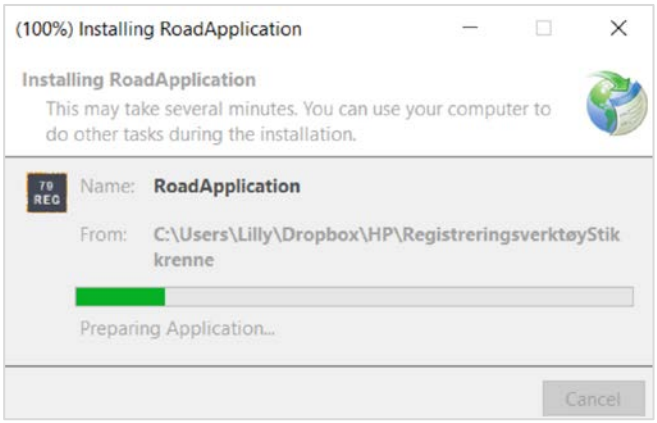


Figure 13-4: Installing the application

The installation will only take a few seconds, and the start page of the culvert registration tool will open as shown in Figure 13-5.

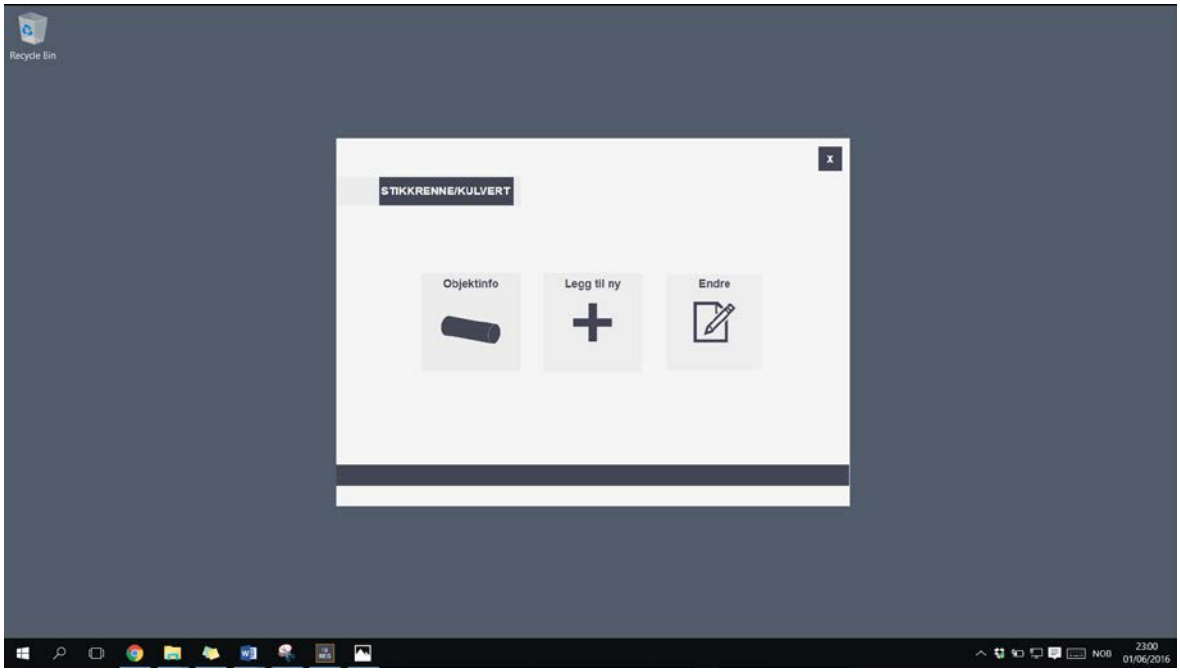


Figure 13-5: Desktop with the culvert registration tool running

When the application is installed, the program can be started directly from the application file. The application is now available along with the other apps on the computer.

Discussion

Suggestions for future implementations:

In this thesis, the GNSS aspect has mainly been a part of the research phase, where the concept of GNSS was discussed. An interactive map within the application handled the positioning. For future implementations, a GNSS should be included in the application and the maps should be directed to the location of the GNSS.

Another suggestion to future work is to expand the functionality of the registration to NRDB writing-API. The culvert registration tool is limited to registration of culvert properties, because of the late release of the developer tool. Further step would be to write the accurate location and associated road objects.

Further development of the application is to create a similar registration procedure for manholes, which will make the application more adapted for road object serving purposes within the same field.

Conclusion

Regarding the position data, the culvert specifications require an accuracy of 20 cm. An accurate measurement system such as used today, is needed to ensure a precise and reliable position measurement.

The application prototype is developed with a user interface, which aims to be user friendly when doing fieldwork. Meaning, fewer distractions, bigger fonts, buttons and other elements within the application. The focus when developing the application, is for it to be easy to operate by employees who are not working directly in the field of road object registration.

To speed up the registration process, based on the scenario of doing quick registration work in the field, the registration procedure starts by giving the user the possibility to set values to the properties categorized as *required* and *conditional*. When reaching the overview, the user can choose to add more properties if needed and add associated objects of the culvert.

Within the application, the position of the culverts is set by using the map service Leaflet, with map tiles provided by Kartverket. The interactive map provided by Leaflet makes it possible to retrieve map boundaries to use for the request of culverts within specific areas. One of the functionalities of Leaflet is the possibility to place the marker on a map and get the geographical coordinates accordingly, such as longitude and latitude. The geographical coordinates are used to give the culvert desired position data. Existing culverts are also pinned to the map using red/orange circles.

Road information needed within the application is retrieved from NRDB using the NRDB API. Information used in the application is data such as road object description, existing culverts within a specific area, road references based on latitude and longitude and possible property input values.

A user manual for the culvert registration tool has been developed to provide the user with a systematic review of the programs content and functionality.

References/literature

1. Statens vegvesen. *Nasjonal vegdatabank (NVDB)*. 2015 19/02/15 [cited 2016 18/01]; Available from: http://www.vegvesen.no/fag/Teknologi/Nasjonal+vegdatabank.
2. difi. *Norsk lisens for offentlige data (NLOD)*. 2016 [cited 2016 20/01]; Available from: http://data.norge.no/nlod/no/1.0.
3. Statens vegvesen. *The National Road DataBase*. 2015 25/07/2015 [cited 2016 18/01]; Available from: http://www.vegvesen.no/en/Professional/Roads+and+transport/National+Road+Data+Bank+NRDB.
4. Statens vegvesen. *Vegkart*. 2015 19/02/2015 [cited 2016 18/01]; Available from: http://www.vegvesen.no/fag/Teknologi/Nasjonal+vegdatabank/Kart/vegkart?fast_title=Vegkart.
5. Statens Vegvesen. *Datakatalogen*. 2016 22/04/2016 [cited 2016 07/05]; Available from: http://www.vegvesen.no/Fag/Teknologi/Nasjonal+vegdatabank/Datakatalogen.
6. difi. *Nasjonal vegdatabank API*. 2016 [cited 2016 20/01]; Available from: http://data.norge.no/data/statens-vegvesen/nasjonal-vegdatabank-api.
7. Vegdata.no. *Utviklerutgave av skrive-apiet tilgjengelig på docker-hub*. [cited 2016 12/05]; Available from: http://www.vegdata.no/2016/03/09/utviklerutgave-av-skrive-apiet-tilgjengelig-pa-docker-hub/.
8. TRIONA. *Vegviseren*. [cited 2016 23/02]; Available from: https://triona.no/produkter_og_tjenester/produkter/vegviseren/.
9. VegReg.no. *VegVIseren*. 2010 [cited 2016 22/02]; Available from: www.vegreg.no/konferanse-2010/VegViseren.ppt.
10. Novatel. *Step 1 - Satellites*. [cited 2016 02/02]; Available from: http://www.novatel.com/an-introduction-to-gnss/chapter-2-basic-gnss-concepts/step-1-satellites/.
11. GPS. *What is GPS?* [cited 2016 02/02]; Available from: http://www.gps.gov/systems/gps/.
12. GPS. *Augmentation Systems*. [cited 2016 30/05]; Available from: http://www.gps.gov/systems/augmentations/.
13. Novatel. *GLONASS*. [cited 2016 11/05]; Available from: http://www.novatel.com/an-introduction-to-gnss/chapter-3-satellite-systems/glonass.
14. Wikipedia contributors. *Geographic coordinate system*. 13/02/2016 [cited 2016 24/02]; Available from: https://en.wikipedia.org/wiki/Geographic_coordinate_system#UTM_and_UPS_systems.
15. Wikipedia contributors. *Earth ellipsoid*. 09/01/2016 [cited 2016 24/02]; Available from: https://en.wikipedia.org/w/index.php?title=Earth_ellipsoid&oldid=698950709.
16. NGA Office of GEOINT Sciences. *THE UNIVERSAL GRID SYSTEM*. 2007 [cited 2016 24/02]; Available from: http://earth-

info.nga.mil/GandG/coordsys/images/utm_mgrs_images/universal_grid_basics_20070319.pdf.

17. Art of Directional Drilling, *UTM grid*. 2015.
18. Wikipedia contributors. *World Geodetic System*. 15/02/2016 [cited 2016 25/02]; Available from: https://en.wikipedia.org/w/index.php?title=World_Geodetic_System&oldid=705064412.
19. Dick, Ø. *WGS84*. 2007 14/02/2009 [cited 2016 30/05]; Available from: https://snl.no/WGS84.
20. Kartverket. *CPOS*. 11/01/2016 [cited 2016 17/02]; Available from: http://kartverket.no/Posisjonstjenester/CPOS/.
21. GPS World. *What Exactly Is GPS NMEA Data?* [cited 2016 8/02]; Available from: http://gpsworld.com/what-exactly-is-gps-nmea-data/.
22. EngineersGarage. *GPS Receivers and NMEA Standards*. [cited 2016 09/02]; Available from: http://www.engineersgarage.com/tutorials/gps-receivers-nmea-standards.
23. Vegdirektoratet, *Nasjonalt vegreferansesystem*. 2010: Statens vegvesen.
24. Halvorsen, H.-P. *Software Testing*. [cited 2016 01/06]; Available from: http://home.hit.no/~hansha/documents/software/software_development/topics/resources/Software%20Testing%20Overview.pdf.
25. Eeles, P. *Capturing Architectural Requirements*. 2005 [cited 2016 20/05]; Available from: http://www.ibm.com/developerworks/rational/library/4706.html.
26. WPF tutorial. *Visual Studio Express*. [cited 2016 12/05]; Available from: http://www.wpf-tutorial.com/getting-started/visual-studio-express/.
27. WPF tutorial. *What is WPF?* [cited 2016 12/05]; Available from: http://www.wpf-tutorial.com/about-wpf/what-is-wpf/.
28. WPF tutorial. *WPF vs. WinForms*. [cited 2016 12/05]; Available from: http://www.wpf-tutorial.com/about-wpf/wpf-vs-winforms/.
29. Microsoft. *Try it: Create a WPF user control*. [cited 2016 22/05]; Available from: https://msdn.microsoft.com/en-us/library/cc294992.aspx.
30. Statens vegvesen. *NVDB REST-API BETA*. [cited 2016 10/05]; Available from: https://www.vegvesen.no/nvdb/api/dokumentasjon/.
31. Microsoft. *WebRequest Class*. [cited 2016 10/05]; Available from: https://msdn.microsoft.com/en-us/library/system.net.webrequest(v=vs.110).aspx.
32. Microsoft. *HttpWebRequest.Accept Property*. [cited 2016 10/05]; Available from: https://msdn.microsoft.com/en-us/library/system.net.httpwebrequest.accept(v=vs.110).aspx.
33. Microsoft. *HttpWebRequest.GetResponse Method ()*. [cited 2016 10/05]; Available from: https://msdn.microsoft.com/en-us/library/system.net.httpwebrequest.getresponse(v=vs.110).aspx.
34. Microsoft. *StreamReader Class*. [cited 2016 10/05]; Available from: https://msdn.microsoft.com/en-us/library/system.io.streamreader(v=vs.110).aspx
35. Wikipedia contributors. *Leaflet (software)*. 22/04/2016 [cited 2016 08/05]; Available from: https://en.wikipedia.org/w/index.php?title=Leaflet_(software)&oldid=716537342.

36. Kartverket. *Vilkår for bruk*. 20/11/15 [cited 2016 30/05]; Available from: http://www.kartverket.no/Kart/Gratis-kartdata/Lisens/.

37. Statens vegvesen. *NVDB API SKRIV*. [cited 2016 30/05]; Available from: http://192.168.99.100:8080/nvdb/apiskriv/doc/.

38. Microsoft. *ASP.NET Cookies Overview*. [cited 2016 09/05]; Available from: https://msdn.microsoft.com/en-us/library/ms178194.aspx.

Appendix A - Task Description

Telemark University College

Faculty of Technology

FMH606 Master's Thesis

Title: Development of Prototype for Registration of Road Data

TUC supervisor: Hans-Petter Halvorsen

External partner: Statens vegvesen (Norwegian Public Roads Administration)

Task description:

Analyze, design and development of a prototype for an application that can be used to register typical road data, e.g., detection and registration of objects (road signs, etc.) that are mounted on and along roads.

The system application needs to be able to communicate with the NVDB API for registration of data. NVDB (in Norwegian: "Nasjonal vegdatabank") is National road data bank (database) where all information about all Norwegian roads are stored.

Use of GPS location will also be of interest.

Some challenges may be: The system should be able work under all kind of weather conditions, accuracy of the GPS location is important and challenges with data communication along Norway's roads.

Task background:

The Norwegian Public Roads Administration (Norwegian: Statens vegvesen) is a Norwegian government agency responsible for the state and county public roads in the country. This includes planning, construction and operation of the state and county road networks, driver training and licensing, vehicle inspection and subsidies to car ferries.

Student category: SCE students

Practical arrangements: It is an advantage if the student can speak Norwegian

Signatures:

Supervisor (date and signature): Hans-Petter Halvorsen 2/2 2016

Students (date and signature): Lilly Eirin Eikehaug 29/01/2016

Adress: Kjølnes ring 56, NO-3918 Porsgrunn, Norway. **Phone:** 35 57 50 00. **Fax:** 35 55 75 47.

Appendix B - Product Specification

Produktspesifikasjon

Datagruppe:	**1**	**Alle**
Vegobjekttype:	**1.4320**	**Stikkrenne/Kulvert (ID=79)**
Datakatalog versjon:	**2.04 - 733**	
Sist endret:	**2014-10-20**	
Definisjon:	Rør for vanngjennomløp på tvers av vegen (evnt. på tvers av tilgrensende avkjørsel) med maks lysåpning 2,5 meter. Stikkrenne/kulvert har åpent innløp og/eller utløp. Stikkrenne/kulvert kan ha inn- og utløpskonstruksjoner som kummer og støtteskjold. Merknad: Inntil videre registrere stikkrenner med bruksområde biologisk mangfold eller landbruk som vanlig stikkrenne. Dette blir endret på i senere versjon av Datakatalogen.	
Kommentar:		

Oppdateringslogg

Dato	Datakatalog versjon	Endringer
2014-09-11		Første versjon
2014-10-20		Ny tegneregel
2015-03-19	2.04 - 733	Krav til nøyaktighet endret fra 10 cm til 20 cm

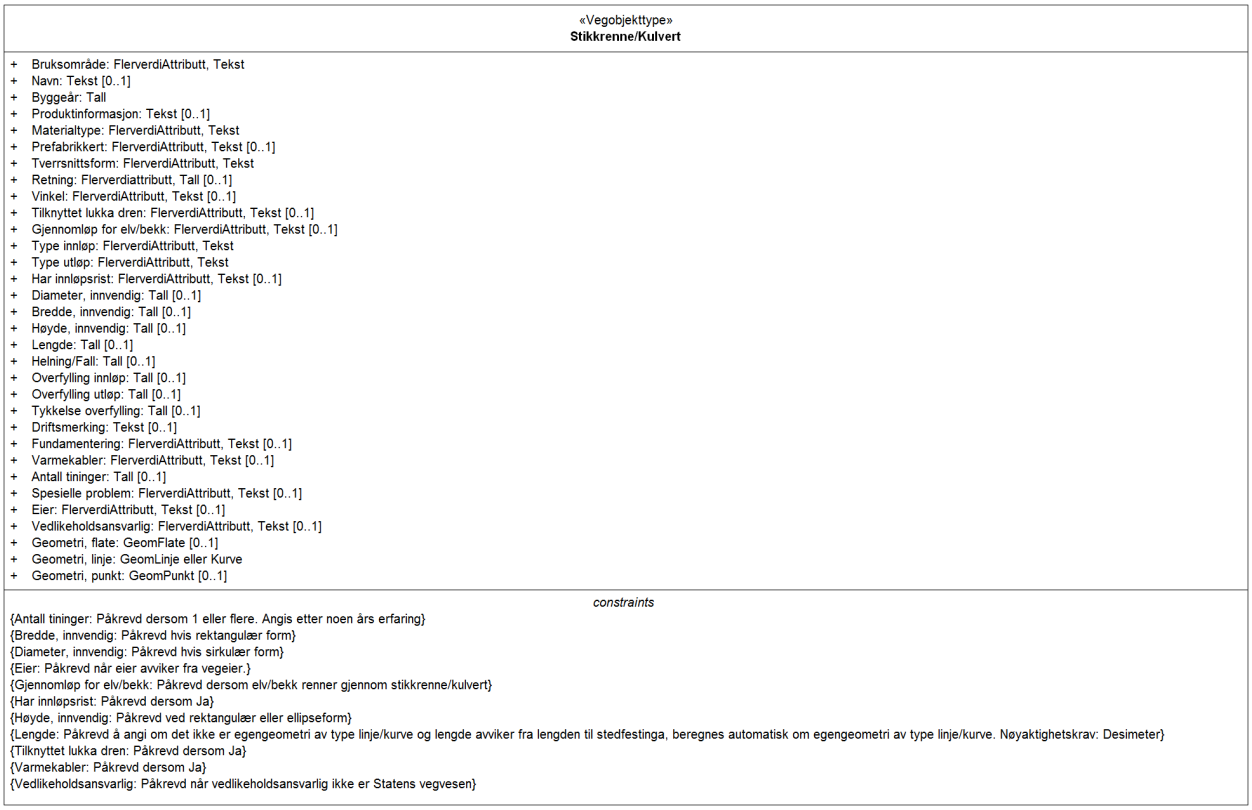
1. Kjente bruksområder og behov

Her listes kjente bruksområder for dataene, og hvilke behov disse bruksområdene har.

Bruksområde	Behov	Eksempel
Drift og vedlikehold	Bruksområde, diameter, varmekabel, type inn- og utløp, antall tininger mm.	Lete fram stikkrenne som er skjult av snø, finne egenskaper for stikkrenne med problemer.
Planlegging	Beliggenhet, bruksområde, diameter, helning	

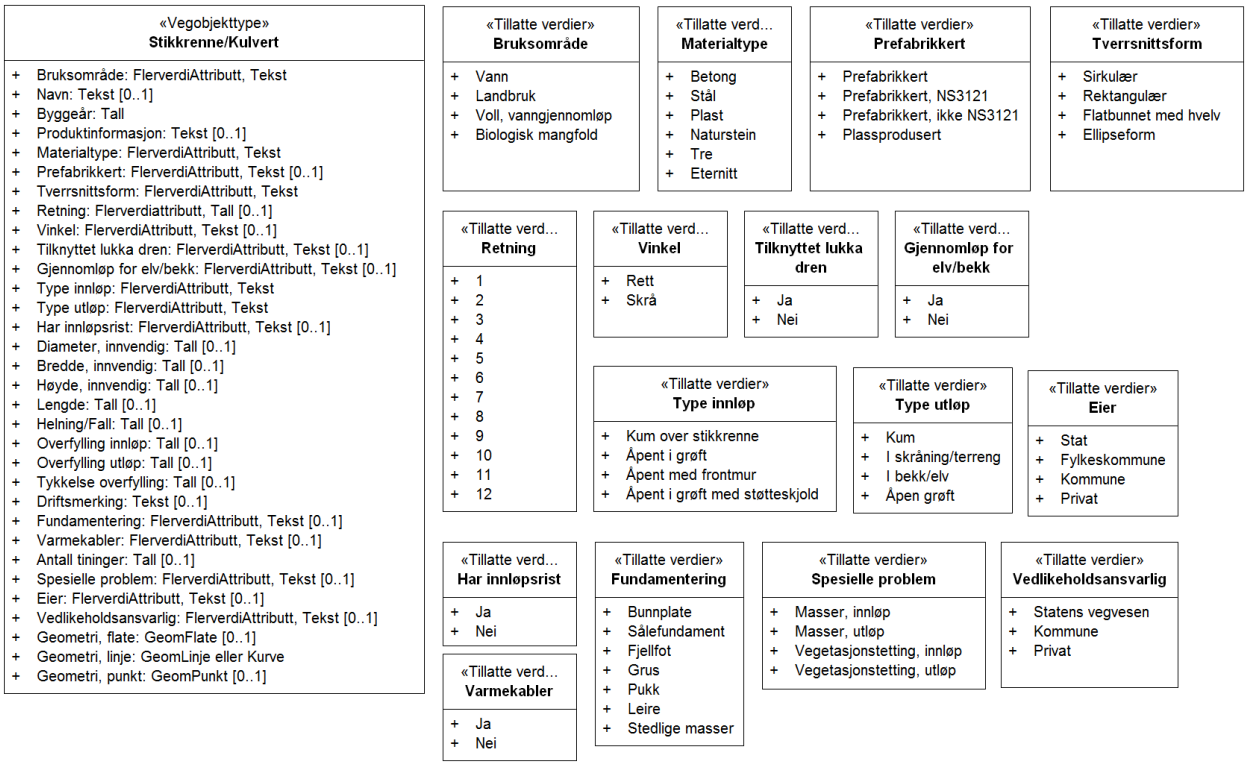
2. Innhold og struktur

2.1 UML-skjema



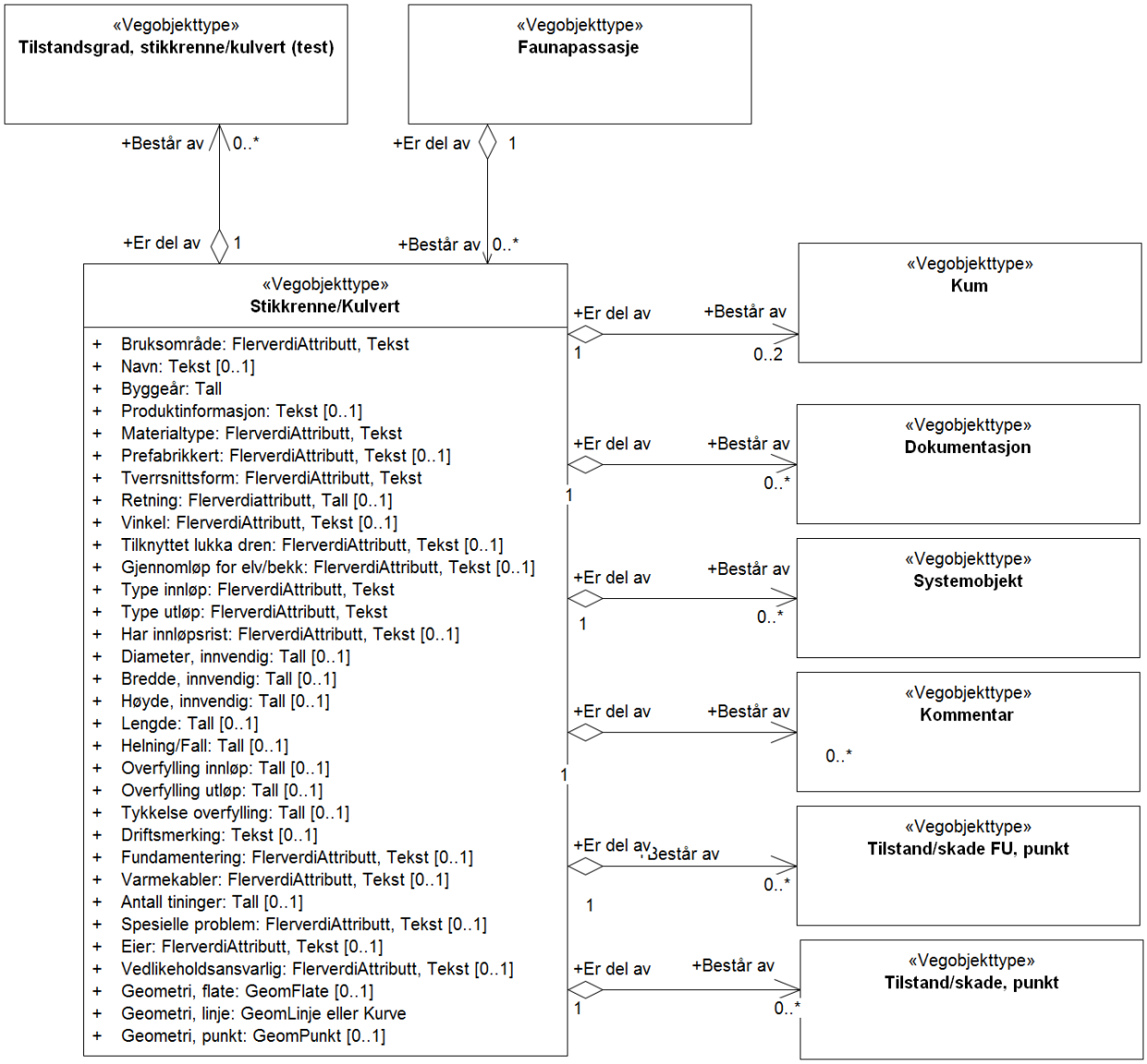
Figur 1: UML-skjema med betingelser

Tillatte verdier



Figur 2: UML-skjema Tillatte verdier

UML-skjema med assosiasjoner



Figur 3: UML-skjema med assosiasjoner

2.2 Beskrivelse av vegobjekttype og tilhørende egenskapstyper

Vegobjekttype

Navn vegobjekttype:	Stikkrenne/Kulvert
Definisjon:	Rør for vanngjennomløp på tvers av vegen (evnt. på tvers av tilgrensende avkjørsel) med maks lysåpning 2,5 meter. Stikkrenne/kulvert har åpent innløp og/eller utløp. Stikkrenne/kulvert kan ha inn- og utløpskonstruksjoner som kummer og støtteskjold. Merknad: Inntil videre registrere stikkrenner med bruksområde biologisk mangfold eller landbruk som vanlig stikkrenne. Dette blir endret på i senere versjon av Datakatalogen.
Representasjon i vegnettet:	punkt
Sideposisjon:	Relevant
Kjørefelt:	Ikke relevant

Egenskapstyper - geometri - tillatte verdier

Tabellen beskriver hver egenskapstype tilhørende vegobjekttypen.

Egenskapstypenavn:	Navn på egenskapstypen(attributten)
Verdi:	Viser tillatte verdier for gitt egenskapstype

Datatype:	Viser datatype og feltlengde.T=Tekst, H=Heltall, D=desimaltall, DATO=dato, FVH/FVT=kodeliste som kan inneholde lister med heltall eller tekster. Heltall bak datatypen viser antall tegn/siffer.
Betingelse:	Angir egenskapstypens viktighet A = Absolutt påkrevd. Krav om verdi for å kunne lagre forekomst P = Påkrevd - Krav om verdi, men mulig å lagre forekomst uten verdi B = Betinget - Krav om verdi når gitte forutsentninger inntreffer O = Opsjonell - Ikke krav om verdi S = Opsjonell spesialinformasjon - Benyttes for spesielle formål. Ikke krav om verdi U = Utgår - Egenskapstype vil bli tatt ut av NVDB. Det skal ikke registreres nye data til denne. Slike egenskaper får prefiks 'Utgår_'
Beskrivelse:	Viser definisjon av egenskapstype, samt eventuell merknad knyttet til registrering av data

Standard egenskapstyper

Egenskapstypenavn Tillatte verdier	**Datatype**	**Betingelse**	**Beskrivelse**	**ID**
Bruksområde	FVT 20	P	Angir hva stikkrenne kulvert primært brukes til.	6981
Vann			Gjennomløp for å transportere vann på tvers av vegen.	9114
Voll, vanngjennomløp			Gjennomløp for å lede vann gjennom voll	15880
Landbruk			Gjennomløp under veg som benyttes i forbindelse med landbruk	9115
Biologisk mangfold			Gjennomløp for å hindre at veg begrenser biologisk mangfold	9116
Navn	T 50	O	Angir navn knyttet til stikkrenne/kulvert	6980
Byggeår	H 4	P	Angir hvilket år stikkrenna ble bygd.	4556
Produktinformasjon	T 50	O	Angir produktinformasjon, først og fremst om selve røret	3111
Materialtype	FVT 40	P	Angir materialtype	6983
Betong				9125
Stål				9126
Plast				9127
Naturstein				9128
Tre				9124
Eternitt				9129
Prefabrikkert	FVT 30	O	Angir om gjennomløp er plassprodusert eller prefabrikkert. Bare aktuelt for stikkrenne/kulvert av betong	6985
Prefabrikkert			Stikkrenne/kulvert er prefabrikkert	9135
Prefabrikkert, NS3121			Stikkrenne/kulvert er prefabrikkert, består av utskiftbare moduler som er i henhold til NS3121	17377
Prefabrikkert, ikke NS3121			Stikkrenne/kulvert er prefabrikert, består av ikke standardiserte moduler	17382
Plassprodusert			Stikkrenne er støpt på stedet	9137
Tverrsnittsform	FVT 30	P	Angir hvilken type tverrsnitt gjennomløpskonstruksjon har.	6984
Sirkulær				9130
Rektangulær				9131
Flatbunnet med hvelv				9132
Ellipseform				9133
Retning	FVH 2	O	Angir hvilken retning i forhold til metrering vegobjektet har. Angir klokkeretning som vannet renner i, 12 angir at vannet renner parallelt med vegen i metreringsretningen	2049
1				3732
2				3741
3				3748
4				3018
5				3779
6				3782
7				3933
8				3934
9				3935

10				3936
11				3937
12				3938
Vinkel	FVT 10	S	Angir om vinkel mellom stikkrenna og veg som stikkrenna krysser er rett eller skrå.	2123
Rett				2465
Skrå				2381
Tilknyttet lukka dren	FVT 3	B	Angir om stikkrenne er tilknytta lukka drenering. Vannet ledes inn i et lukket dreneringssystem. Merknad: Påkrevd dersom Ja	1941
Ja				3533
Nei				3571
Gjennomløp for elv/bekk	FVT 3	B	Angir om elv/bekk renner gjennom stikkrenne/kulvert Merknad: Påkrevd dersom elv/bekk renner gjennom stikkrenne/kulvert	10223
Ja				16700
Nei				16701
Type innløp	FVT 50	P	Angir hvilken type innløp det er i ei stikkrenne	1939
Kum over stikkrenne				2925
Åpent i grøft			Vann renner inn direkte fra åpen grøft.	11744
Åpent med frontmur				16699
Åpent i grøft med støtteskjold				16761
Type utløp	FVT 50	P	Angir hvilken type utløp det er i ei stikkrenne	1940
Kum			Vann ledes til kum	2927
I skråning/terreng			Vann ledes ut i skråning eller ut i terreng	2928
I bekk/elv			Vann ledes ut i bekk/elv	2929
Åpen grøft			Vann ledes til åpen grøft. Merknad: Ofte aktuelt i forbindelse med stikkrenner under avkjørsel.	11655
Har innløpsrist	FVT 3	B	Angir om det er innløpsrist i tilknytning til vegobjektet Merknad: Påkrevd dersom Ja	1923
Ja				3531
Nei				3569
Diameter, innvendig	H 4 (mm)	B	Angir innvendig diameter av gjennomløp. Benyttes fortrinnsvis for sirkulære tverrsnitt Merknad: Påkrevd hvis sirkulær form	3113
Bredde, innvendig	H 4 (mm)	B	Angir innvendig bredde av gjennomløpskonstruksjon. Ikke aktuell for sirkulære tverrsnitt Merknad: Påkrevd hvis rektangulær form	4548
Høyde, innvendig	H 4 (mm)	B	Angir innvendig høyde av gjennomløpskonstruksjon. Tar ikke hensyn til evt. gjennfylling i bunn av konstruksjon.. Merknad: Påkrevd ved rektangulær eller ellipseform	4549
Lengde	D 6 (m)	B	Angir lengde av vegobjektet Merknad: Påkrevd å angi om det ikke er egengeometri av type linje/kurve og lengde avviker fra lengden til stedfestinga, beregnes automatisk om egengeometri av type linje/kurve. Nøyaktighetskrav: Desimeter	1323
Helning/Fall	D 6	O	Angir fall på stikkrenne. Angis alltid som positiv verdi.	3112
Overfylling innløp	D 4 (m)	O	Angir tykkelsen på overfylling ved innløp. Det vil si tykkelse fra topp av stikkrenne til topp dekke.	10224
Overfylling utløp	D 4 (m)	O	Angir tykkelsen på overfylling ved utløp. Det vil si tykkelse fra topp av stikkrenne til topp dekke.	10225
Tykkelse overfylling	D 4 (m)	S	Angir tykkelse overfylling av rørledning. Det vil si gjennomsnittlig tykkelse fra topp av rørledning til topp dekke.	3115
Driftsmerking	T 50	O	Gir unikt navn/id for objektet	10481
Fundamentering	FVT 999	S	Angir hvordan stikkrenne/kulvert er fundamentert	6982
Bunnplate				9117
Sålefundament				9118
Fjellfot				9119

Grus				9120
Pukk				9121
Leire				9122
Stedlige masser				9123
Varmekabler	FVT 3	B	Angir om det er varmekabler eller ikke i tilknytning til vegobjektet Merknad: Påkrevd dersom Ja	1832
Ja				3524
Nei				3562
Antall tininger	H 2 (stk)	B	Angir hvor mange ganger stikkrenna vanligvis må tines i løpet av en vinter Merknad: Påkrevd dersom 1 eller flere. Angis etter noen års erfaring	1942
Spesielle problem	FVT 50	O	Angir eventuelle spesielle problem knyttet til stikkrennen. Dette er problem som stadig gjentar seg.	4562
Masser, innløp				5470
Masser, utløp				5471
Vegetasjonstetting, innløp				5472
Vegetasjonstetting, utløp				5473
Eier	FVT 50	B	Angir hvem som er eier av vegobjektet. Merknad: Påkrevd når eier avviker fra vegeier.	7996
Stat				10262
Fylkeskommune				10724
Kommune				10326
Privat				10390
Vedlikeholdsansvarlig	FVT 50	B	Angir hvem som er ansvarlig for vedlikehold Merknad: Påkrevd når vedlikeholdsansvarlig ikke er Statens vegvesen	8060
Statens vegvesen				10454
Kommune				10532
Privat				10610
Utgår_Høyde, passasje	H 4 (mm)	U	Angir innvendig høyde når det er tatt hensyn til eventuelle hindringer, f.eks masser i bunn, oppheng i tak. Normalt ikke aktuelt å oppgi for vannggjennomløp.	6979

Geometri egenskapstyper

Egenskapstypenavn	Datatype	Betingelse	Beskrivelse	ID
Geometri, linje	GLK	P	Gir linje/kurve som geometrisk representerer objektet. Merknad: Grunnriss: Senter bunn Stikkrenne/Kulvert med retning fra innløp til utløp - Høydereferanse: Bunn Stikkrenne/Kulvert	5899
Geometri, flate	GF	O	Gir flate/polygon som geometrisk avgrenser området Merknad: Grunnriss: Flate som dekker største bredde for Stikkrenne/Kulvert - Høydereferanse: Bunn Stikkrenne/Kulvert	5902
Geometri, punkt	GP	O	Gir punkt som geometrisk representerer objektet. Merknad: Grunnriss: Senter bunn innløp Stikkrenne/Kulvert. Senter utløp kan brukes der dette er mer hensiktsmessig - Høydereferanse: Bunn Stikkrenne/Kulvert	4780

3. Kvalitetskrav

Kravmatrisen viser de forskjellige krav som stilles til kvalitet på de data som ligger i NVDB for den eller de objekttyper som er behandlet i dette dokumentet. Kravene går på:
Aktualitet = tidsfrist for oppdatering i NVDB i forhold til når fysisk objekt er driftsatt
Fullstendighet = krav til hvor komplett innlegging av objekt eller egenskap skal være
Konsistens = krav til sammenheng mellom objekter av samme eller forskjellig datatype

Kvalitetskravklasser:
1 = Europa- og riksveger
2 = Fylkesveger

3 = Kommunale veger
4 = Private veger og skogsbilveger

Kravene under er gitt i henhold til ny datamodell, og viser maksimalt tillatt avvik

Krav nr	Kvalitets-element	Kvalitetsmål	Rel.vegob type	Egenskap type	Beskrivelse	Kvalitetsklasse			
						1	2	3	4
1778	Fullstendighet, manglende data	Andel manglende data			Alle Stikkrenne/Kulvert skal være registrert	0 %	0 %		
1779	Aktualitet	Tidsperiode, forsinkelse			Data skal være inne i NVDB innen angitt frist	90 dager	90 dager		
1781	Fullstendighet, manglende data	Andel manglende data			Varmekabler skal være angitt dersom Ja	0 %	0 %		
1782	Fullstendighet, manglende data	Andel manglende data			Har innløpsrist skal være angitt dersom Ja	0 %	0 %		
1785	Fullstendighet, manglende data	Andel manglende data			Tilknyttet lukka dren skal være angitt dersom Ja	0 %	0 %		
1786	Fullstendighet, manglende data	Andel manglende data			Antall tininger skal være angitt dersom 1 eller flere. Skal være angitt etter noen års erfaring	0 %	0 %		
1787	Fullstendighet, manglende data	Andel manglende data			Diameter, innvendig skal være angitt hvis sirkulær form	0 %	0 %		
1788	Fullstendighet, manglende data	Andel manglende data			Bredde, innvendig skal være angitt hvis rektangulær form	0 %	0 %		
1789	Fullstendighet, manglende data	Andel manglende data			Høyde, innvendig skal være angitt ved rektangulær eller ellipseform	0 %	0 %		
1797	Fullstendighet, manglende data	Andel manglende data			Vedlikeholdsansvarlig skal være angitt når Vedlikeholdsansvarlig ikke er Statens vegvesen	0 %	0 %		
1839	Fullstendighet, manglende data	Andel manglende data			Eier skal være angitt når eier avviker fra vegeier.	0 %	0 %		
1798	Fullstendighet, manglende data	Andel manglende data			Gjennomløp for elv/bekk skal være angitt dersom elv/bekk renner gjennom Stikkrenne/Kulvert	0 %	0 %		
1780	Fullstendighet, manglende data	Andel manglende data			Lengde skal være angitt på alle objekter	0 %	0 %		
1783	Fullstendighet, manglende data	Andel manglende data			Type innløp skal være angitt på alle objekter	0 %	0 %		
1784	Fullstendighet, manglende data	Andel manglende data			Type utløp skal være angitt på alle objekter	0 %	0 %		
1790	Fullstendighet, manglende data	Andel manglende data			Byggeår skal være angitt på alle objekter	0 %	0 %		
1791	Fullstendighet, manglende data	Andel manglende data			Geometri, linje skal være angitt på alle objekter	0 %	0 %		
1792	Absolutt stedfestings-	Middelverdi av feil i			Avvik i posisjon skal være	20	20		

1792	stedfestings-nøyaktighet	stedfestings-nøyaktighet			innenfor gitt verdi	cm	cm		
1793	Fullstendighet, manglende data	Andel manglende data			Bruksområde skal være angitt på alle objekter	0 %	0 %		
1794	Fullstendighet, manglende data	Andel manglende data			Materialtype skal være angitt på alle objekter	0 %	0 %		
1795	Fullstendighet, manglende data	Andel manglende data			Tverrsnittsform skal være angitt på alle objekter	0 %	0 %		
1811	Konseptuell konsistens	Andel objekter med avvik fra regler i det konseptuelle skjemaet	Kum		Kum som er datterobjekt til Stikkrenne/Kulvert skal ha relevant plassering i forhold til Stikkrenne/Kulvert	0 %	0 %		

4. Innsamlingsregler med eksempler

Nr 1	**Regel:**	Et Stikkrenne/Kulvert-objekt skal registreres for hver Stikkrenne/Kulvert ute langs vegen i henhold til kravmatrisa. Stikkrenne/Kulvert registreres normalt med start innløp. Der det er mer hensiktsmessig for f.eks. tining kan utløp registreres. Stikkrenne/Kulvert inngår ikke i Lukka drenering. Det regnes derfor ikke som stikkrenne der lukka drenering krysser vegen Vi kan ha tilknytning til kum både for innløp og utløp. Videreføring fra vegkroppen fra kum mot rørledning som kan ha egne kummer over eks. dyrka mark Når en stikkrenne blir forlenget på grunn av utvidelse av veg, deles den opp slik at hver del har sine egenskaper. Er det flere rør, registreres det en stikkrenne for hvert rør. Hvis annen metode for tining enn vanlig varmekabel benyttes, f.eks. innlagt rør som det kjøres varmt vann eller damp gjennom settes egenskapen Varmekabler til Ja. Det må da legges inn en Kommentar som beskriver dette.

Doble stikkrenner

Antall tininger : 2
Bruksområde : Vann
Diameter, innvendig : 800 mm
Fundamentering : Pukk
Gjennomløp for elv/bekk : Ja
Har innløpsrist : Nei
Helning/Fall : 1:20
Byggeår : 2006
Lengde : 8 m
Materialtype : Stål
Overfylling innløp : 0.3 m
Overfylling utløp : 0.35 m
Prefabrikkert : Prefabrikkert
Produktinformasjon :
Retning : 9
Spesielle problem : Vegetasjonstetting, innløp
Tilknyttet lukka dren : Nei
Tverrsnittsform : Sirkulær
Tykkelse overfylling : 0.3 m
Type innløp : Åpent i grøft
Type utløp : I bekk/elv
Varmekabler : Nei
Vinkel : Rett

Foto Geir Brekke

Stikkrenne med utstyr for tining

Bildet viser en stikkrenne med fast monterte rør for varmetransporterende væske som brukes for tining av is.

Antall tininger : 5
Bruksområde : Vann
Diameter, innvendig : 800 mm
Fundamentering : Pukk
Gjennomløp for elv/bekk : Ja
Har innløpsrist : Nei
Helning/Fall : 1:15
Byggeår : 2003
Lengde : 12 m
Materialtype : Betong
Overfylling innløp : 0.4 m
Overfylling utløp : 0.35 m
Prefabrikkert : Ja
Produktinformasjon :
Retning : 3
Spesielle problem : Masser innløp
Tilknyttet lukka dren : Nei
Tverrsnittsform : Sirkulær
Tykkelse overfylling : 0.6 m
Type innløp : Åpen i grøft
Type utløp : I bekk/elv
Varmekabler : Ja
Vinkel : Rett

Foto Tomas Rolland

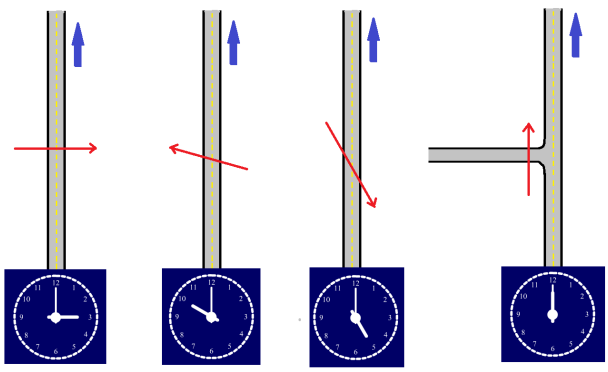
Eldre stikkrenne

Stikkrenne (naust) under RV 720 Mellom Verrabotn og Follafoss i Nord-Trøndelag.

Stikkrenne. Foto: Per Gevik

Retning

Bildet viser eksempler på hvordan Retning : angis.
Blå piler viser vegens metreringsretning.
Rød linje viser hvordan stikkrennen går og pilen hvilken retning vannet renner.
Klokken under viser hvilken verdi som gis inn for retning. Kun hele timer.
Siste eksempel gjelder når stikkrennen refererer til hovedvegen.



Retning for Stikkrenne/Kulvert

Stikkrenne som er forlenget ved utvidet veg

Her vises en gammel stikkrenne av murt stein som er forlenget med et betongrør. I dette tilfellet må det beskrives to stikkrenner som ligger etter hverandre.

Stikkrenne som må deles i to objekter i NVDB

Appendix C - User Manual

Brukerveiledning

Registreringsapplikasjon for stikkrenner

Brukerveiledning for registreringsverktøy for stikkrenner. Applikasjonen er spesielt tilpasset for bruk i felt.

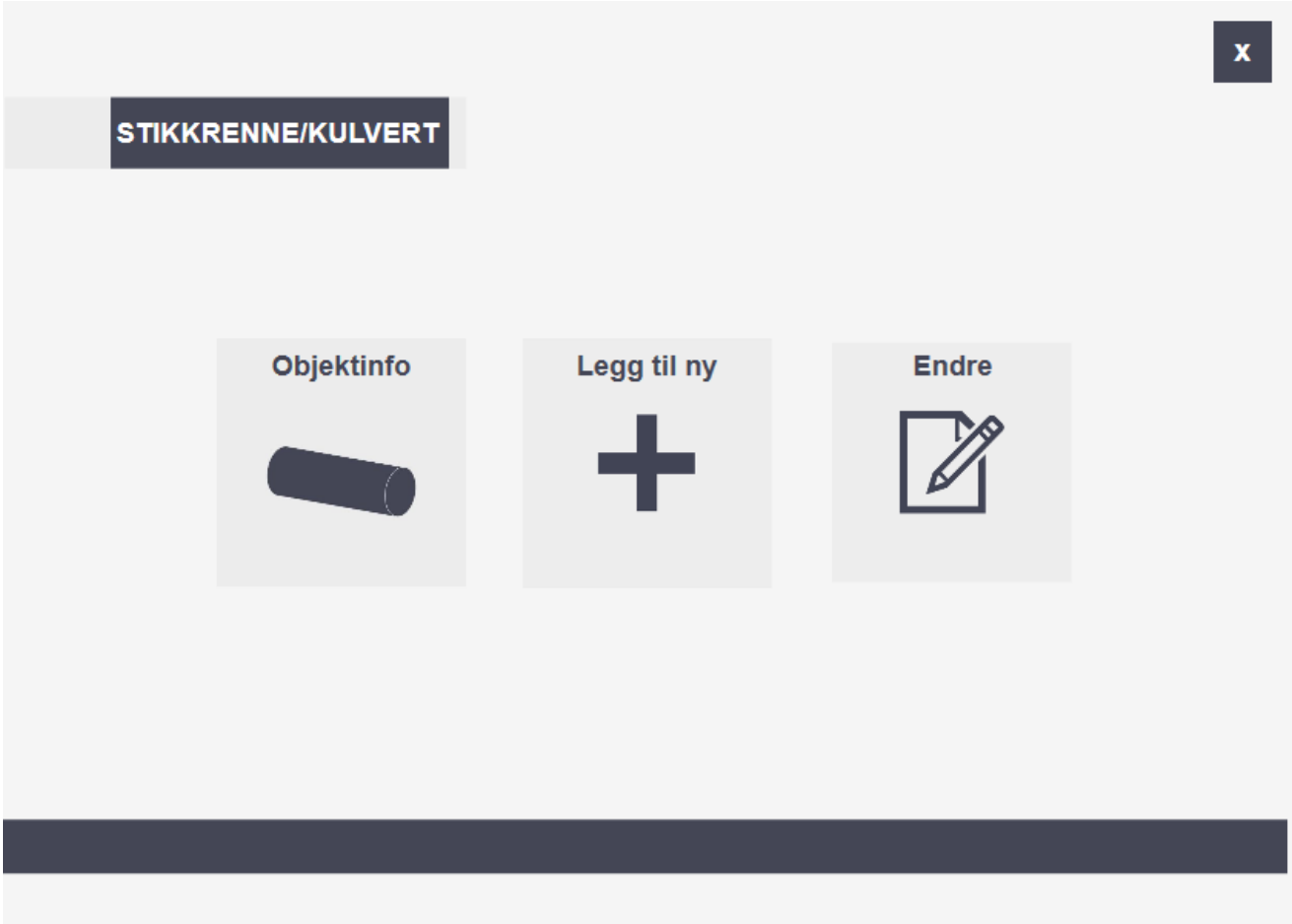
Brukerveiledning

Registreringsapplikasjon for stikkrenner

Innhold

Hovedskjerm

Startbildet ved applikasjonsstart er vist i figuren under.



Startskjermen gir bruker tre navigeringsmuligheter; **Objektinfo**, **Legg til ny** og **Endre**. **Objektinfo** fører til en ny side som gir en overordnet beskrivelse av vegobjektet stikkrenne/kulvert. **Legg til ny** fører videre til en sekvens hvor det kan legges inn påkrevd, betinget og opsjonell informasjon om ny stikkrenne som skal lagres i NVDB. Applikasjonen kan på dette tidspunkt ikke skrive til databasen, men henter all registreringsinformasjon fra NVDB lese-API. Under **Endre** vises stikkrenner i et kartdefinert område. Her kan man undersøke innlagt stikkrenneinformasjon, og ved ønske endre denne informasjonen. (En reel endring i NVDB krever også skriverettigheter). For å avslutte applikasjonen, benyttes **x** øverst i høyre hjørne.

Objektinfo

Figuren under viser knappen **Objektinfo**

Dersom **Objektinfo** velges fra startskjermen, vises skjermbildet i figuren under.

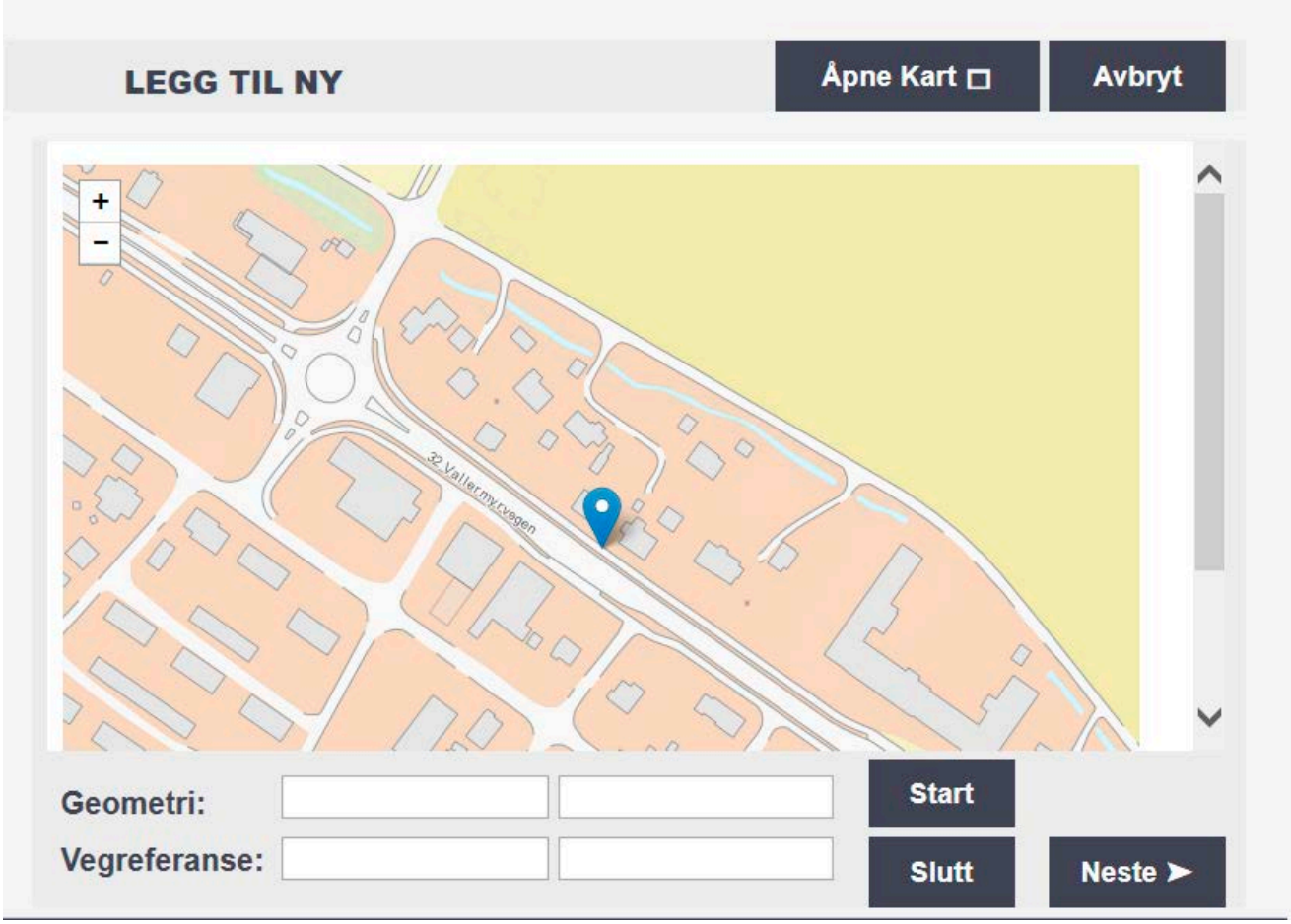
Informasjonen vist på denne siden er hentet direkte fra NVDB som vegobjektets overordnede beskrivelse. Denne beskrivelsen oppdateres ved endring i NVDB. Benytt **Tilbake** for å returnere til startskjermen.

Legg til ny

Figuren under viser knappen **Legg til ny**

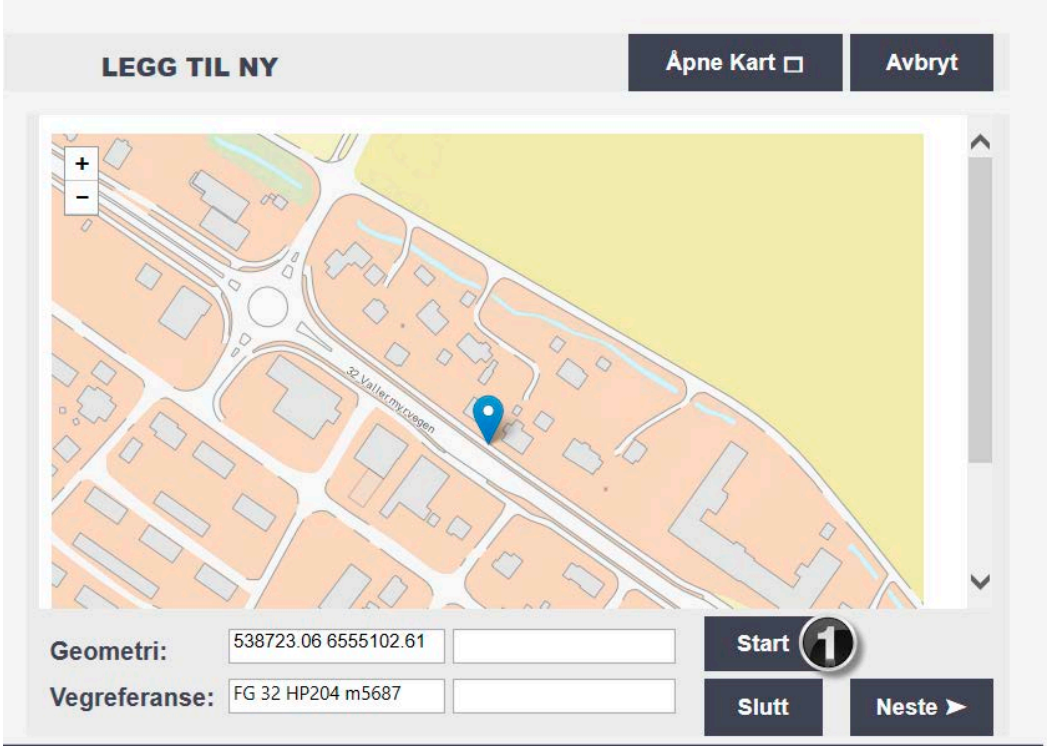
Legg til linjegeometri og vegreferanse

Dersom **Legg til ny** velges fra startskjermen, vises skjermbildet i figuren under.

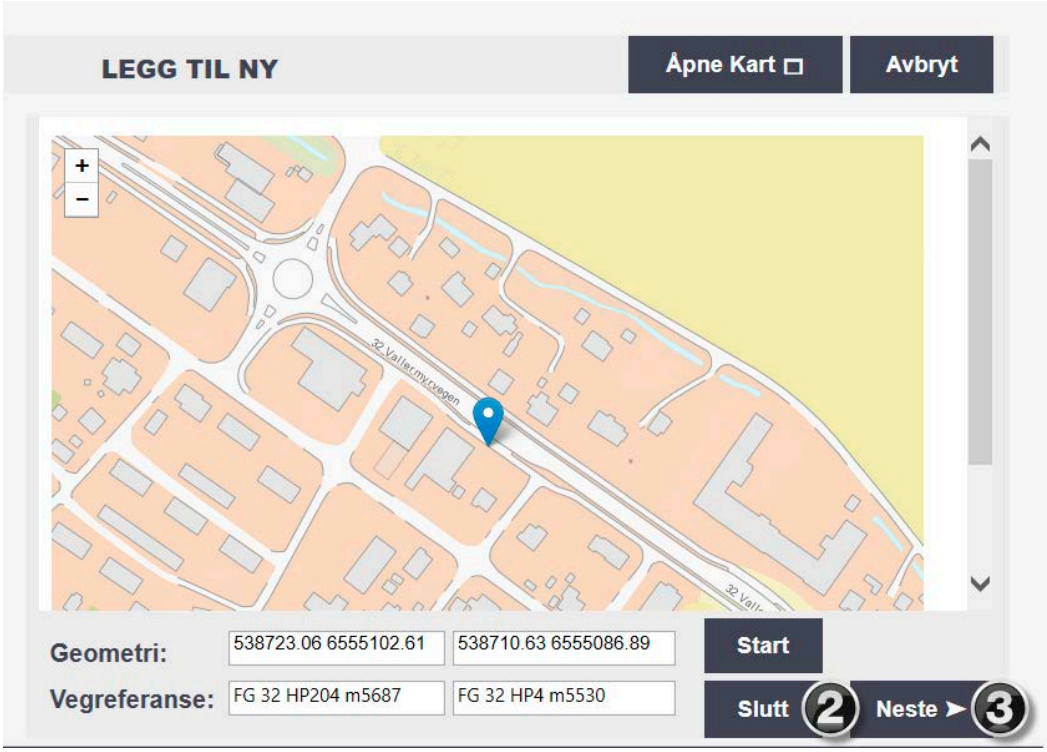


Kartet brukt i denne applikasjonen er tilgjengelig via kartverket, det er derfor nødvendig med internettilkobling. Denne funksjonen gir bruker mulighet til å sette start og stopp posisjon (linjeposisjon) for objektet ved å dra markør til ønsket posisjon. I realiteten vil dette kun foregå via GNSS, ettersom stikkrenne krever en nøyaktighet på 20 cm.

Plasser markør i ønsket posisjon og trykk **Start**, geometri i UTM33 koordinater vises i geometri tekstfelt, og vegreferanse til markøren vises i vegreferanse tekstfeltet.

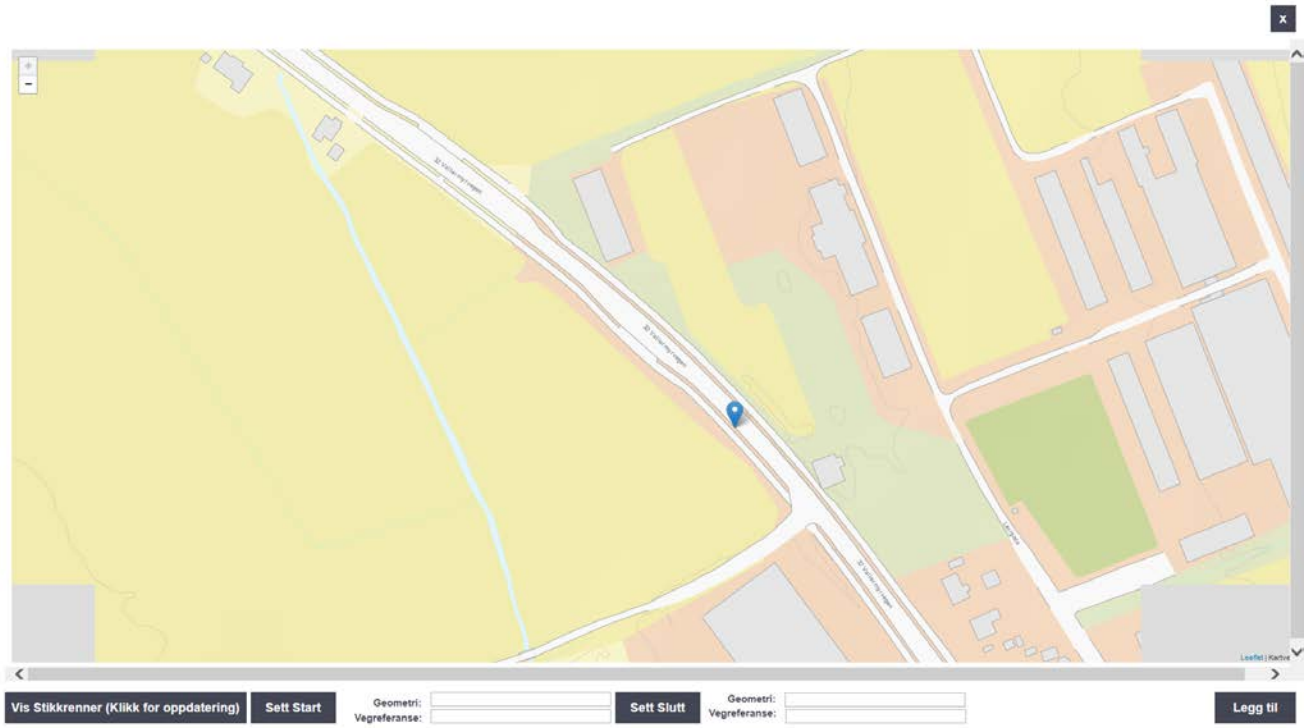


Flytt markøren til sluttposisjon (andre enden av stikkrennen). Velg **Slutt,** posisjon i UTM33 koordinater og vegreferanse vises i tilsvarende tekstbokser.



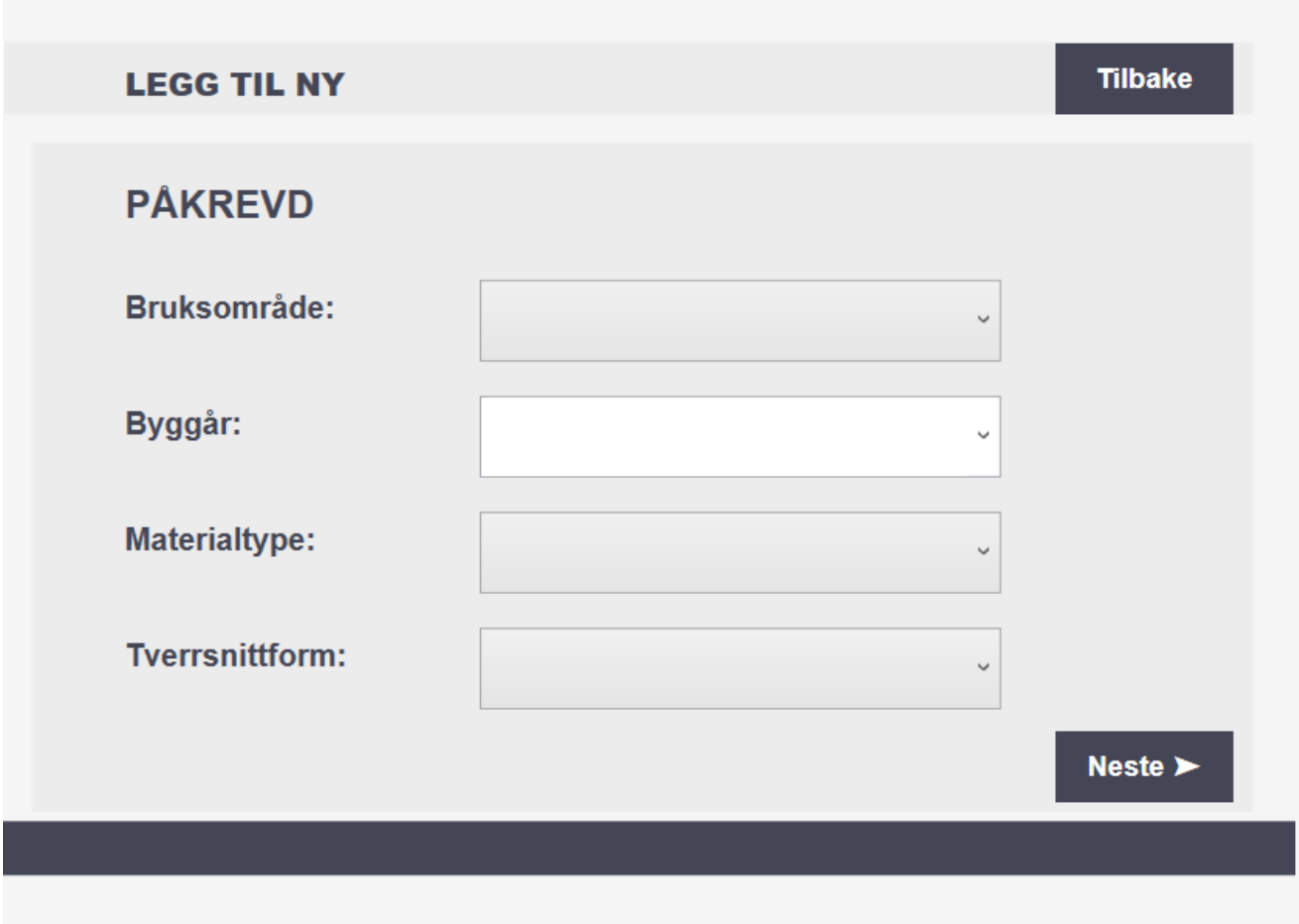
Når start og sluttposisjon er satt og vises i tekstboksene som vist ovenfor, kan bruker gå videre ved å trykke **Neste.**

Dersom bruker ønsker å forstørre kartet, maksimerer **Åpne Kart** vinduet. Vinduet vil da bli seende ut som vist nedenfor, hvor man setter start og slutt posisjon som forklart tidligere. **Vis Stikkrenner** markerer med en sirkel eksiterende stikkrenner i kartområdet. Dersom bruker vil tilbake til normalt vindu, benyttes **x.** Når informasjonen er komplett kan bruker **Legge til** de geografiske data.

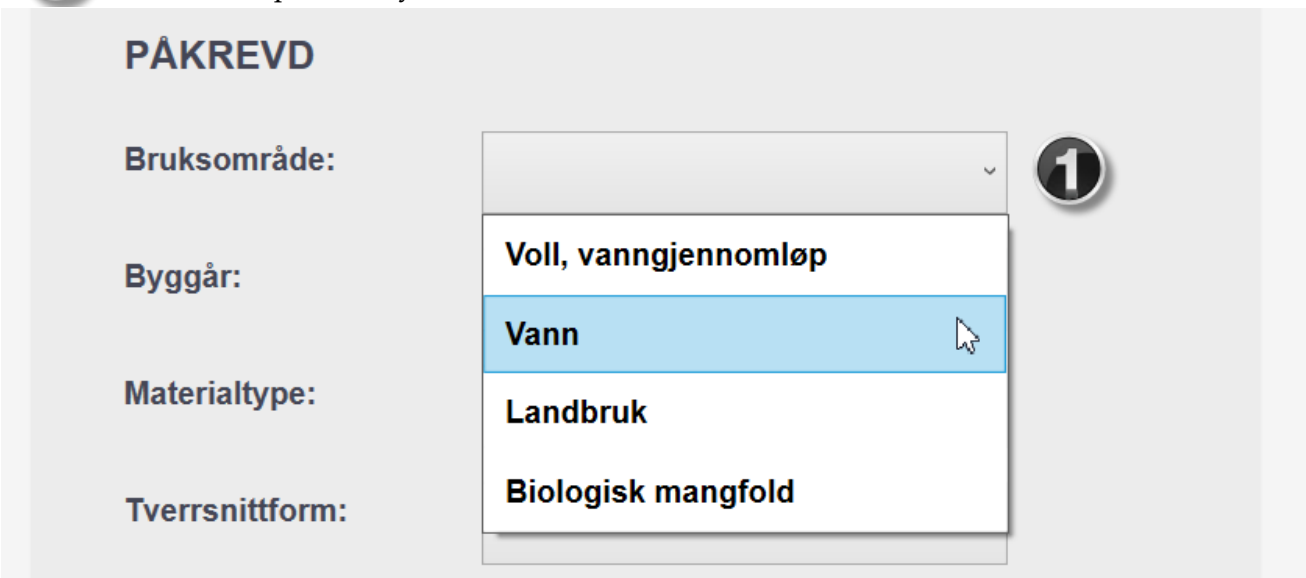


Legg til påkrevd informasjon del 1

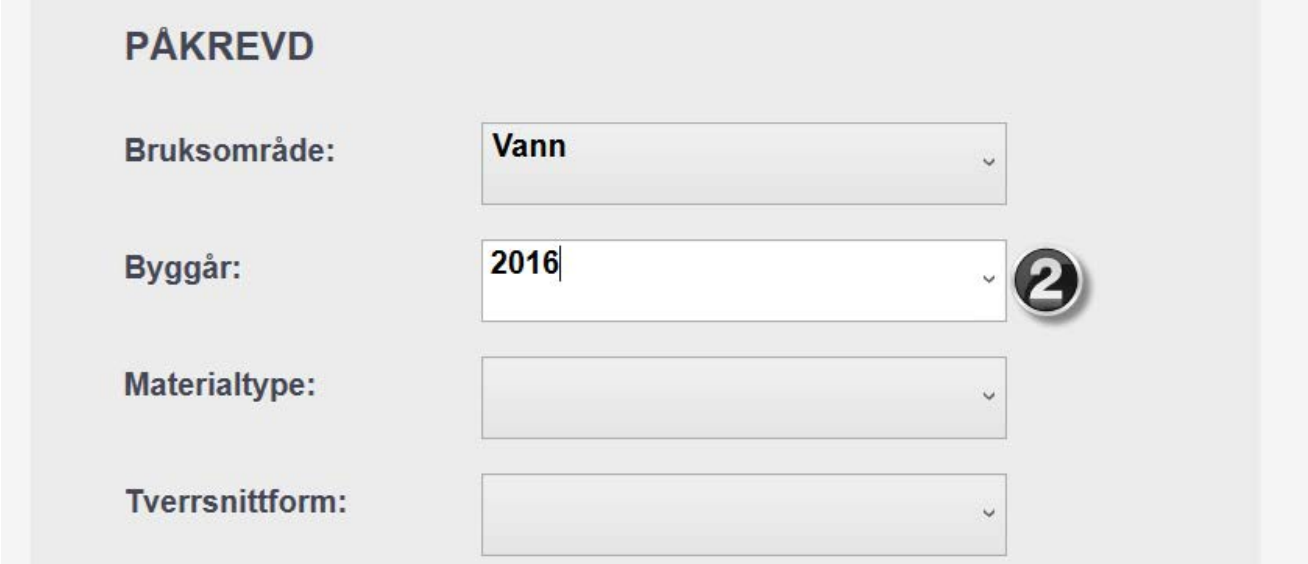
Når posisjon er lagt inn, og **Neste** valgt kommer man til skjermbildet vist i figuren under. Denne siden presenterer første del av påkrevd informasjon som må legges inn ved registrering av ny stikkrenne.



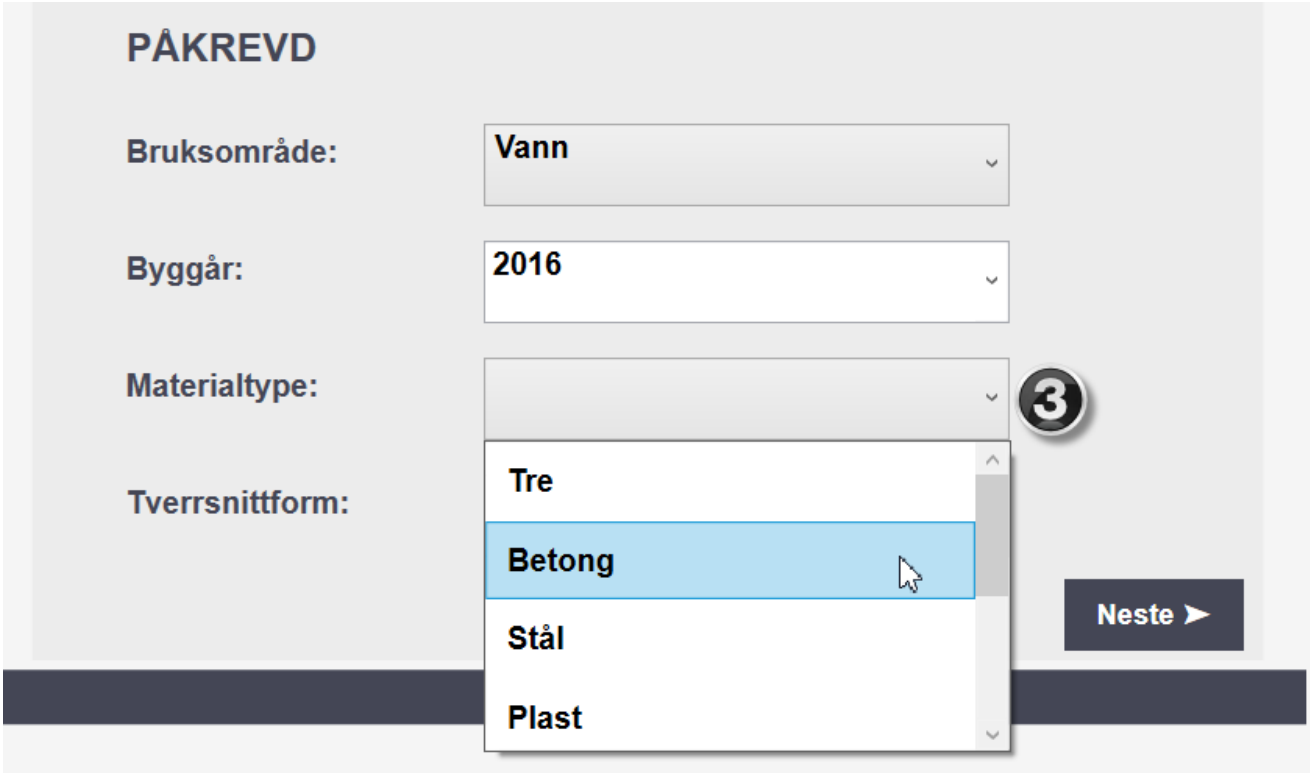
1. Klikk på feltet ved siden av bruksområde for å åpne nedtrekksmenyen. NVDB har definert inputs tilknyttet bruksområde, som vist i listen nedenfor. Klikk ønsket verdi.



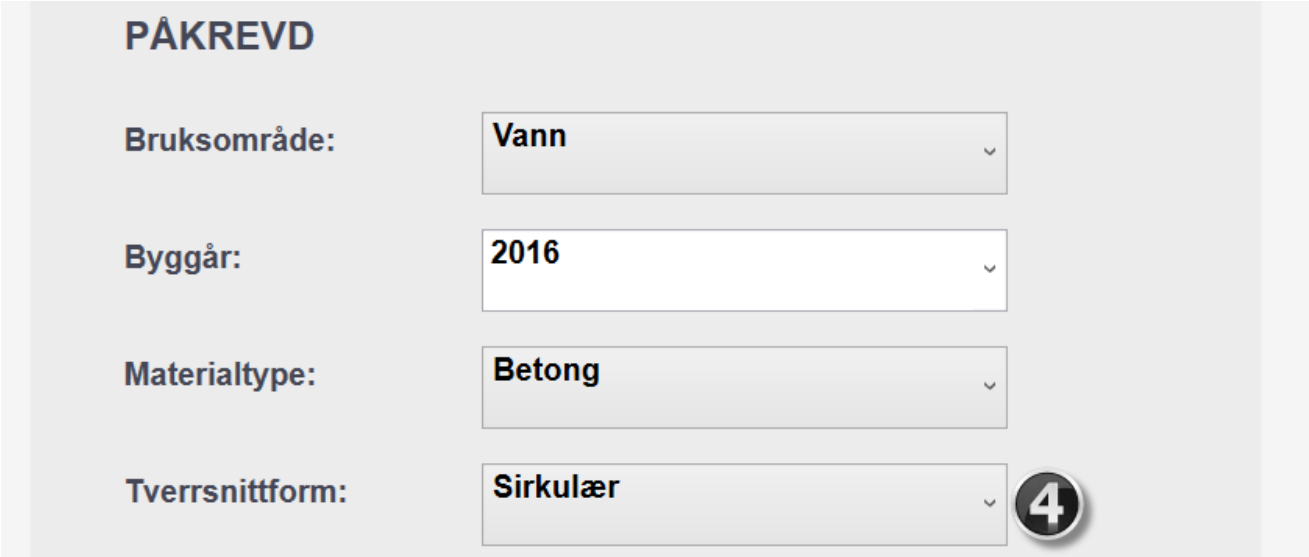
2 I tekstfeltet for Byggår kan numerisk verdi skrives inn ved hjelp av tastatur. Nedtrekksmenyen gir også mulighet for å velge nyere årstall (2013-2016).



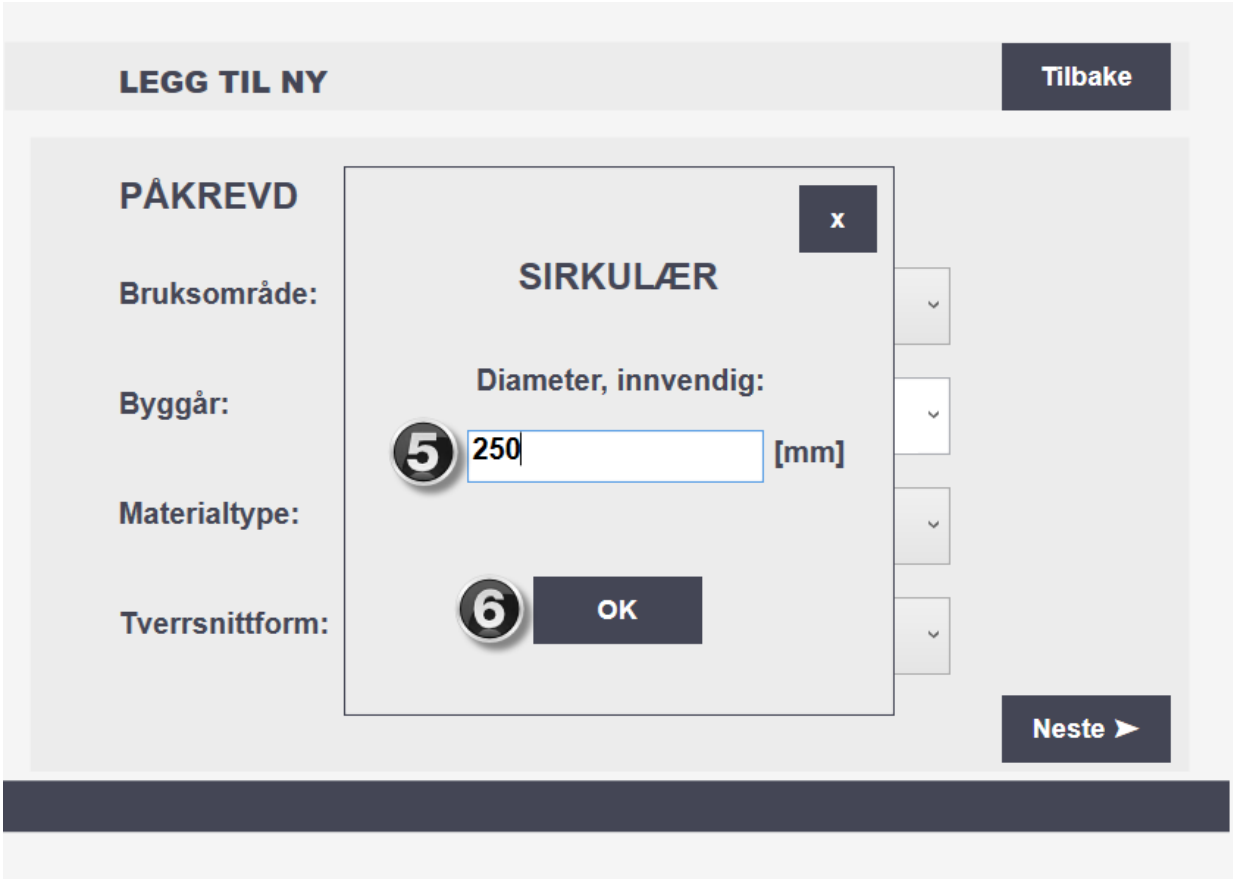
3 Samme prosedyre som i punkt (1), gjelder for materialtype.



4 For Tverrsnittform gjelder også samme prosedyre. Tverrsnittform har betinget informasjon som er tilknyttet valg av form.

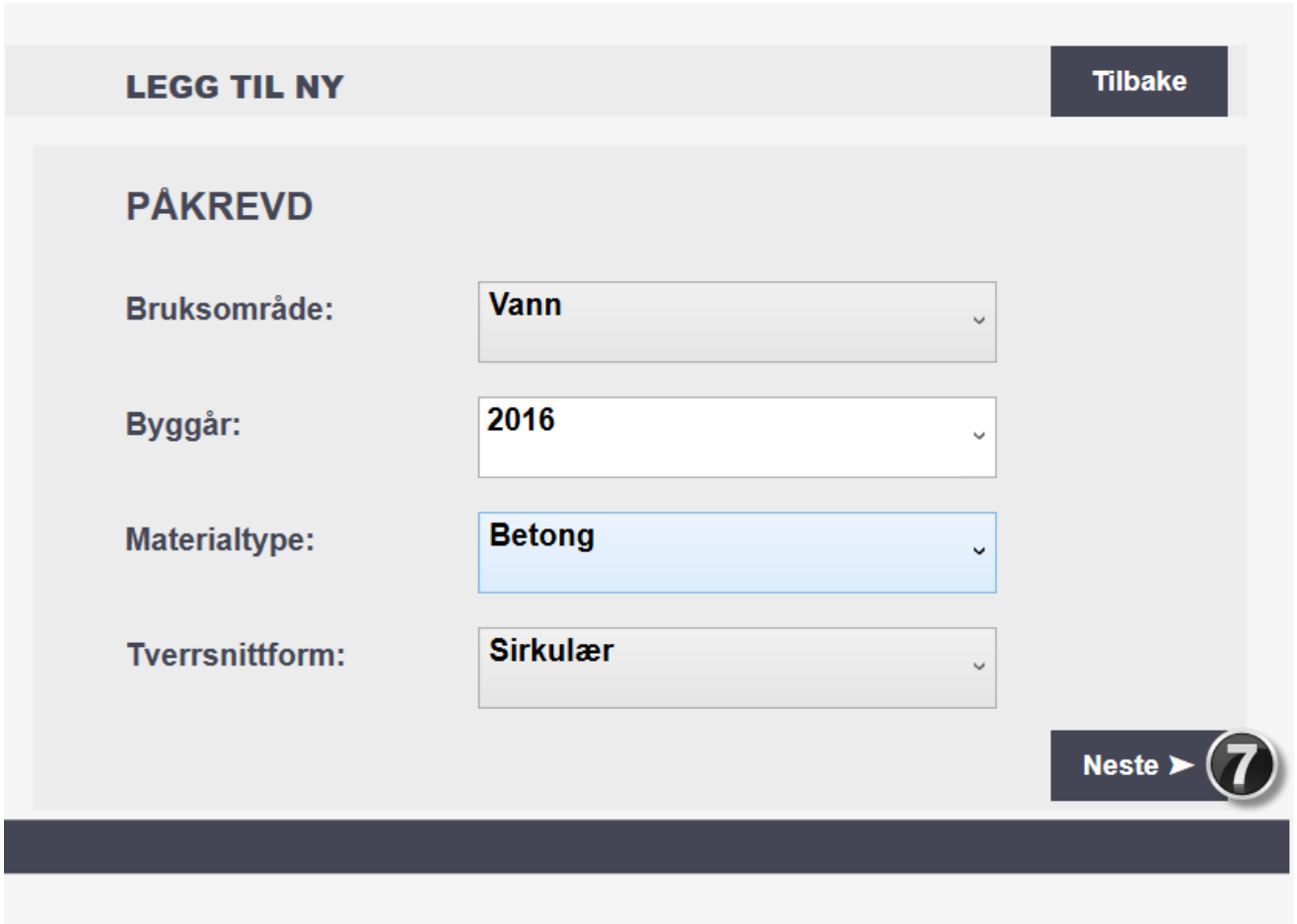


5 6 Dersom valg av tverrsnittform er 'Sirkulær', åpnes boksen vist under. Skriv inn Diameter, innvendig ved hjelp av tastatur og klikk **ok.** Dette skrivefeltet må ha verdi for å fortsette.



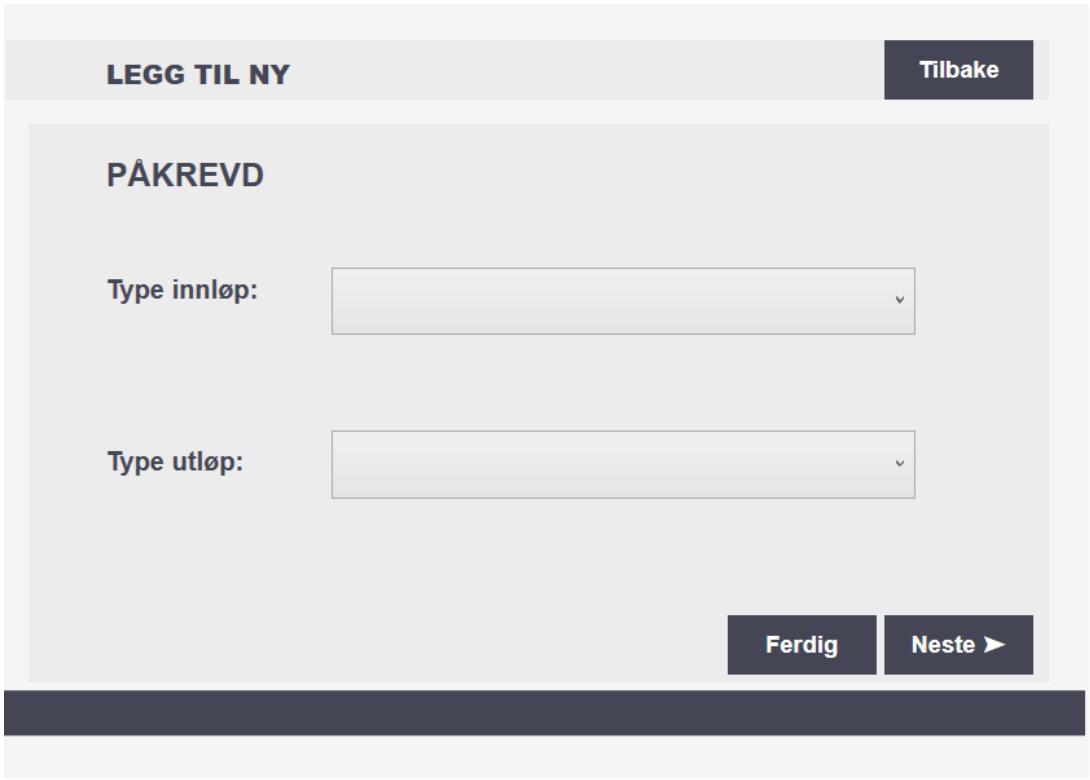
Tilsvarende bokser dukker opp hvis valg av tverrsnittform er 'Rektangulær' eller 'Ellipse', som vist under.

7 Når all informasjonen er lagt inn er det klart for å gå videre til neste sett av påkrevd informasjon, ved å klikke **Neste.**

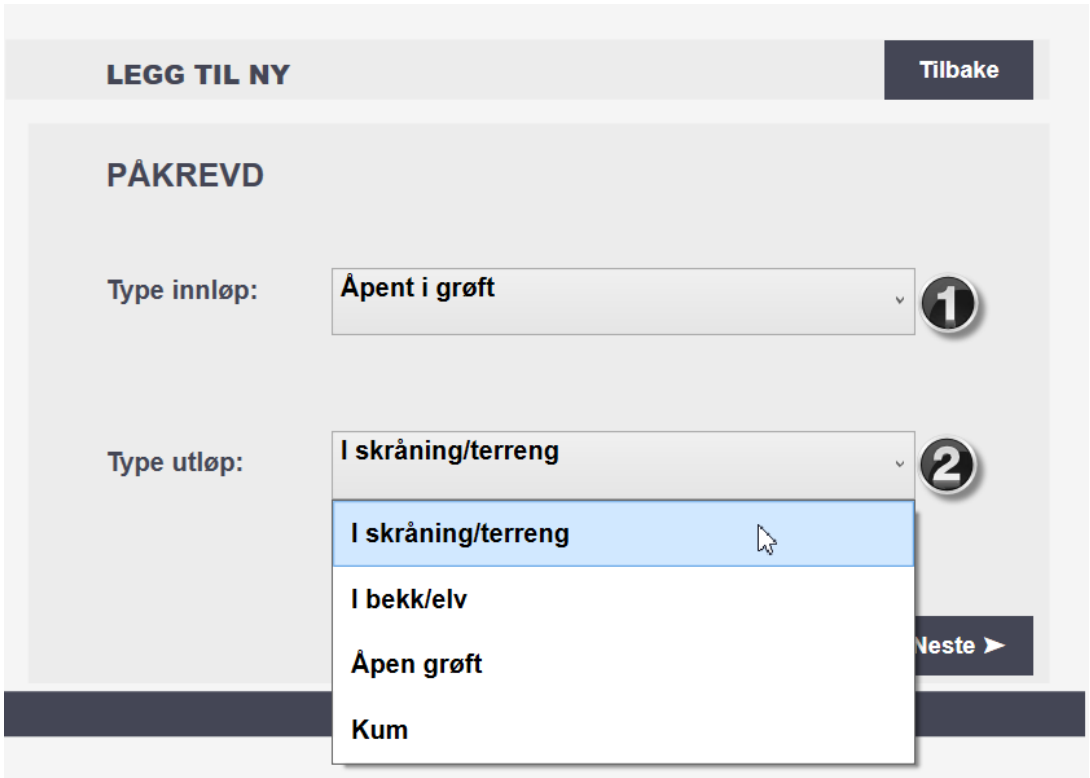


Legg til påkrevd informasjon del 2

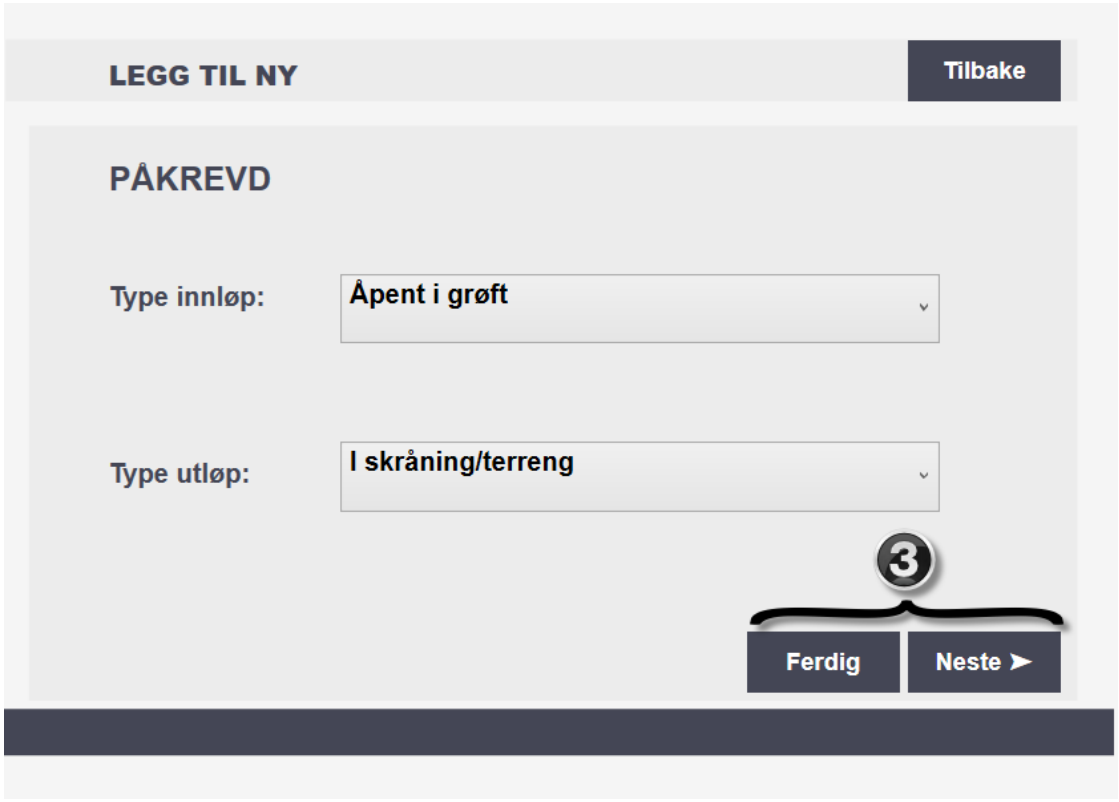
Neste del av påkrevd informasjon er vist i figuren under. Type innløp og type utløp.



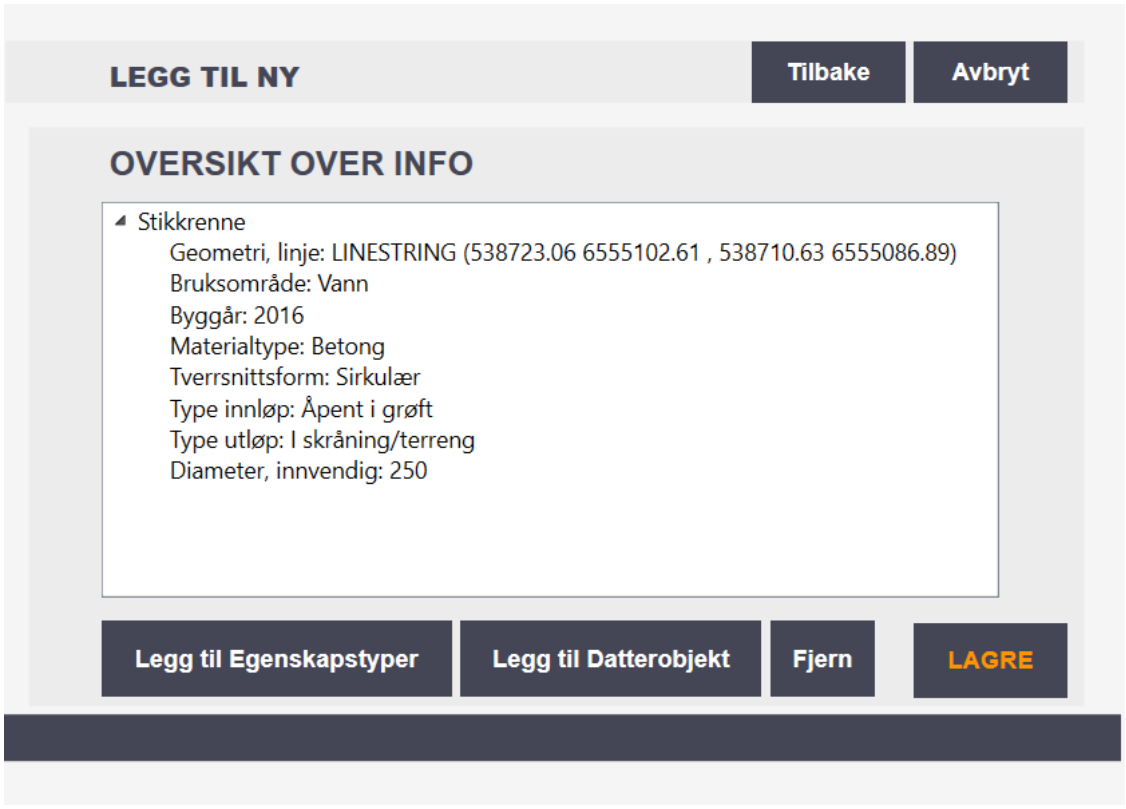
Den samme prosedyren som på forrige side gjelder, velg type innløp og utløp i nedtrykksmenyene.



3 Når informasjonen er satt, står valget mellom å gå rett til oversiktsmenyen ved å velge **Ferdig**, eller gå videre til neste side for betinget informasjon ved å velge **Neste.**



Dersom man velger **Ferdig** kommer man til oversiktssiden vist under.



Legg til betinget informasjon del 1

Betinget informasjon trenger kun å legges inn dersom det er relevant for den aktuelle stikkrennen, eller hvis informasjonen avviker fra standarden. Første del av betinget er vist i figuren under.

1 Velg relevant svar, Ja eller Nei.
Svaralternativet blir markert i oransje farge.

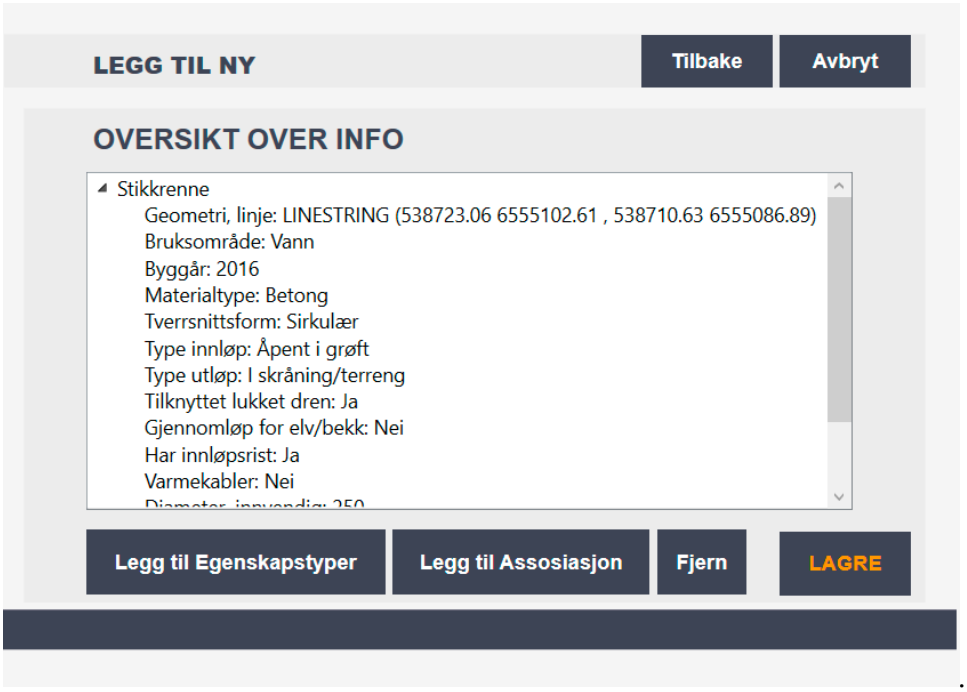
2 Når betinget informasjon er klart, kan man velge **Ferdig** som fører direkte til oversiktssiden, eller **Neste** som fortsetter til neste betinget-side.

Legg til betinget informasjon del 2

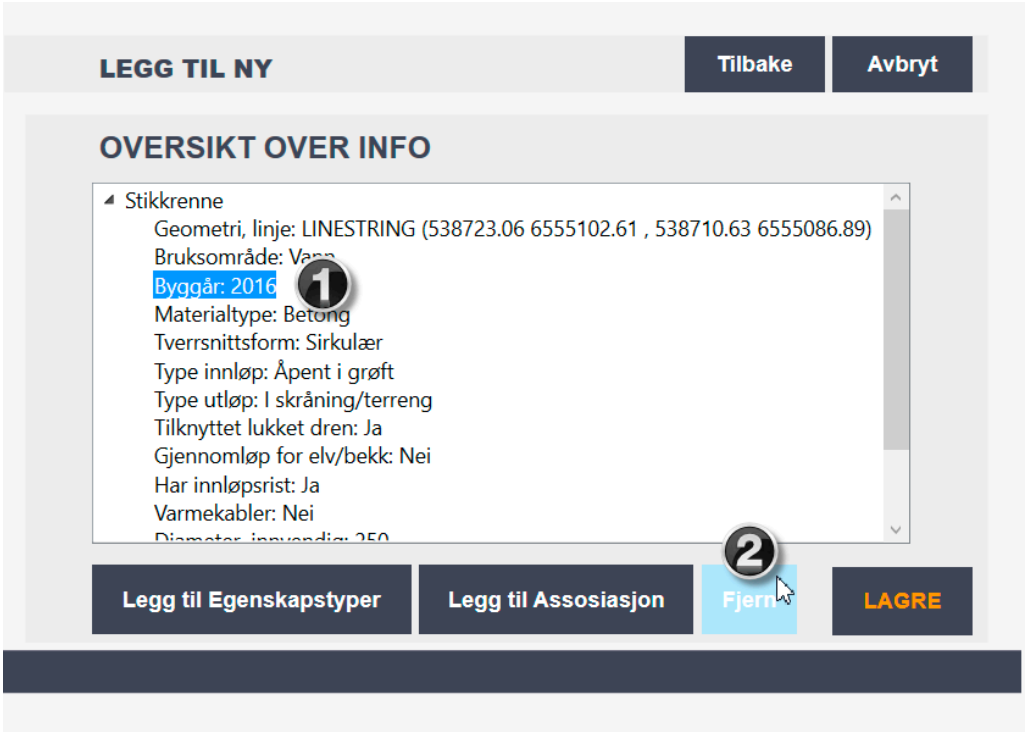
1 2 Betinget del 2 har samme prosedyre som tidligere. Velg verdier fra nedtrekksmenyene og klikk **Neste** for å komme videre til oversikten.

Oversikt over informasjon

Den siste siden i legg til ny sekvensen viser oversikt over innlagt informasjon. Oversikt over info - siden gir mulighet for å **Avbryte** registreringen eller gå **Tilbake** for å gjøre endringer. Det er også muligheter for å **Legge til Egenskapstyper** som ikke allerede har blitt lagt inn, **Legge til Assosiasjon**, og **Fjerne** eventuell feilinformasjon. Når alt ser ok ut, kan bruker **Lagre** stikkrennen.

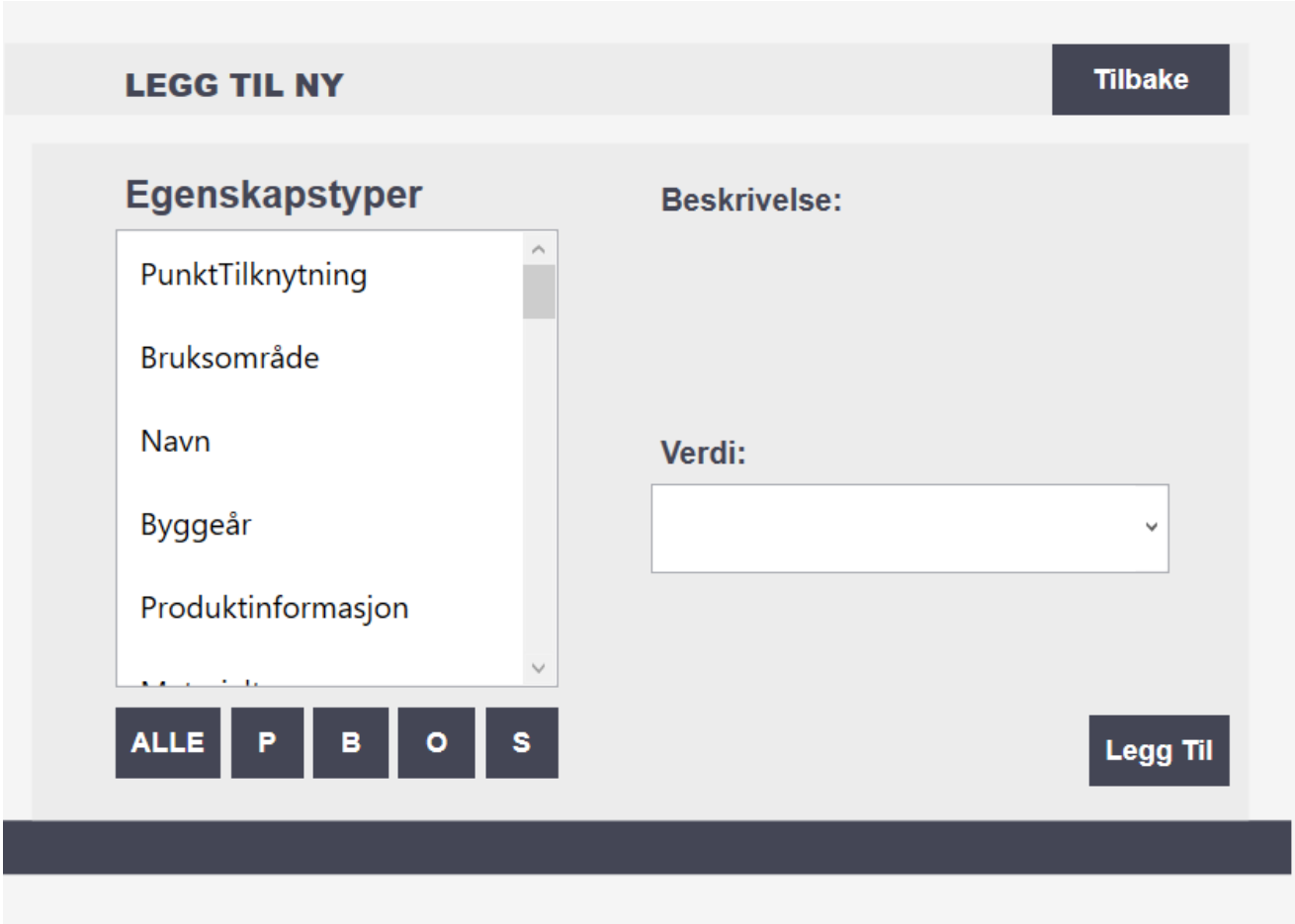


1 2 Dersom det finnes feil i informasjonsoversikten, velges den aktuelle egenskapstypen i menyen, og deretter **Fjern.**



Legg til Egenskapstyper

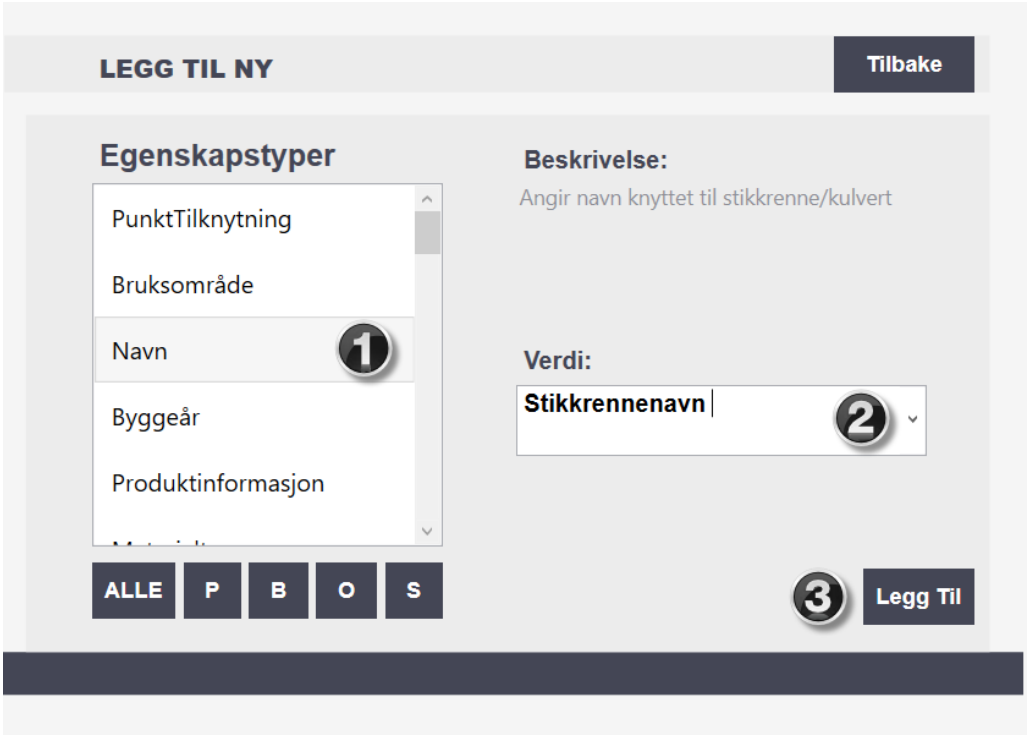
Dersom **Legg til Egenskapstyper** velges på oversiktsinfosiden, vises skjermbildet i figuren under.



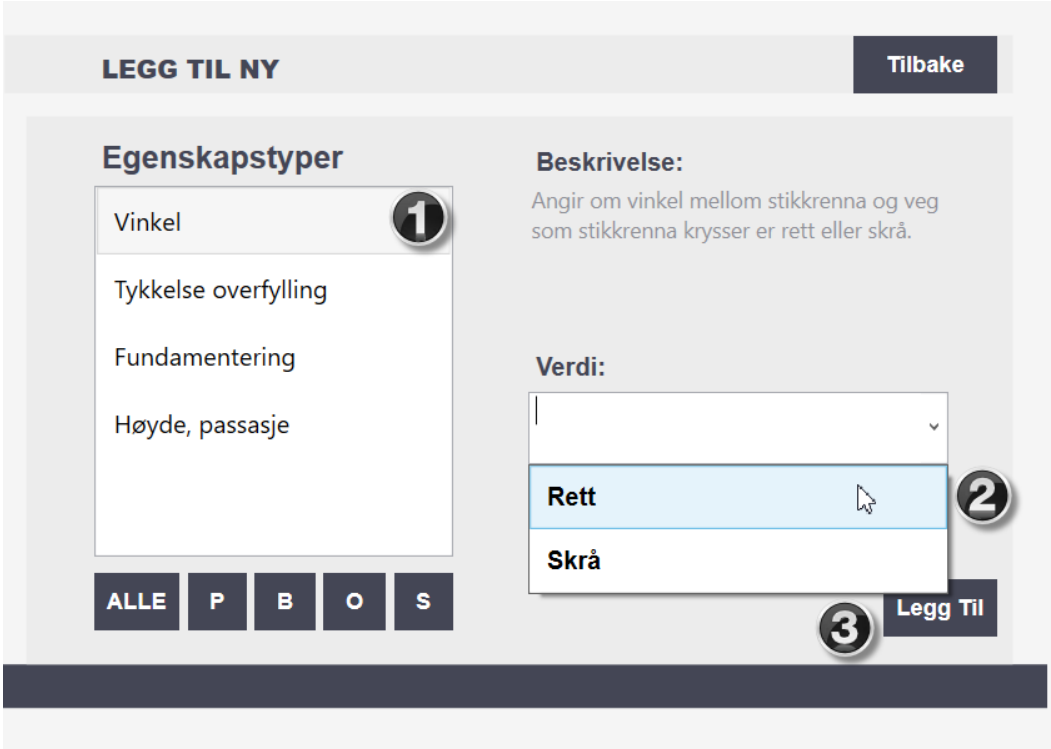
Når siden åpnes, vises alle mulig egenskapstyper for stikkrenne i listen. Dersom man ønsker å se egenskaper med en bestemt viktighet kan man velge (info fra produktspesifikasjon for stikkrenne):

- P– Påkrevd (Krav om verdi, men mulig å lagre forekomst uten verdi)
- B – Betinget (Krav om verdi når gitte forutsetninger inntreffer)
- O – Opsjonell (Ikke krav om verdi)
- S – Opsjonell spesialinformasjon (Benyttes for spesielle formål. Ikke krav om verdi)

1-3 Velg egenskapstype i listen til venstre. Samsvarende beskrivelse vil dukke opp på høyde side under beskrivelse. 'Navn' som har blitt valgt i listen, har ingen innlagte rammeverdier, navnet på stikkrennen skrives derfor inn i tekstruten ved hjelp av tastatur. Når alt er klart, velg **Legg til.**



1-3 Dersom for eksempel egenskapstypen 'Vinkel' velges, finnes tilhørende verdier i nedtrekkmenyen. Når alt er klart, velg **Legg Til**.

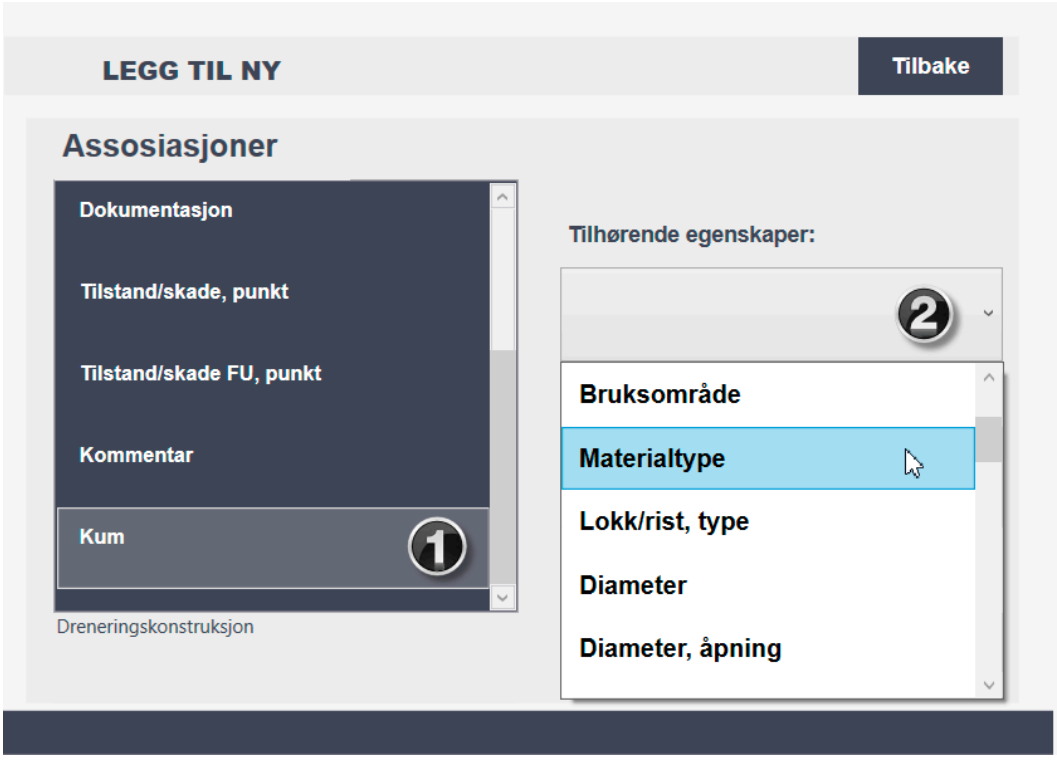


Legg til Assosiasjon

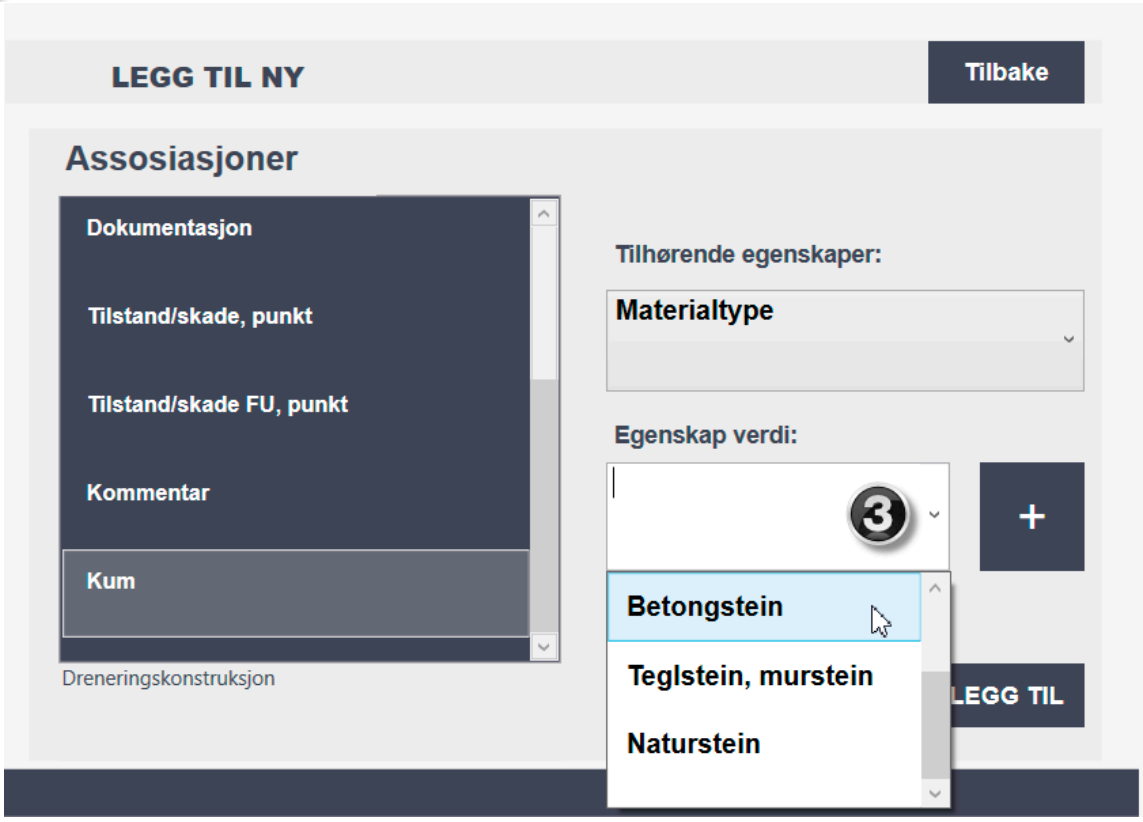
Dersom **Legg til Assosiasjon** velges på oversiktsforsiden, vises skjermbildet i figuren under.

Listen viser stikkrenne/kulvert sine assosierte objekter.

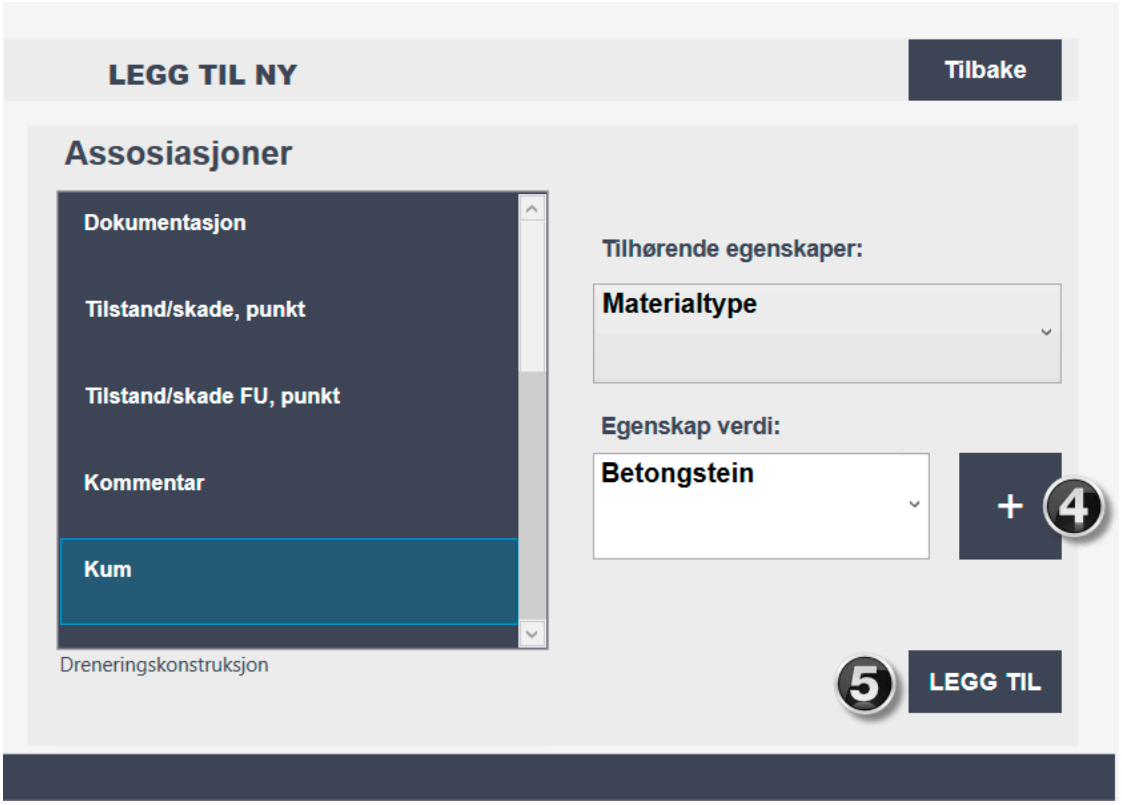
1 2 Velg assosiert objekt i listen til venstre. Beskrivelse av objektet vil dukke opp under listeboksen, og tilhørende egenskaper kan velges fra nedtrekksmenyen til høyre.



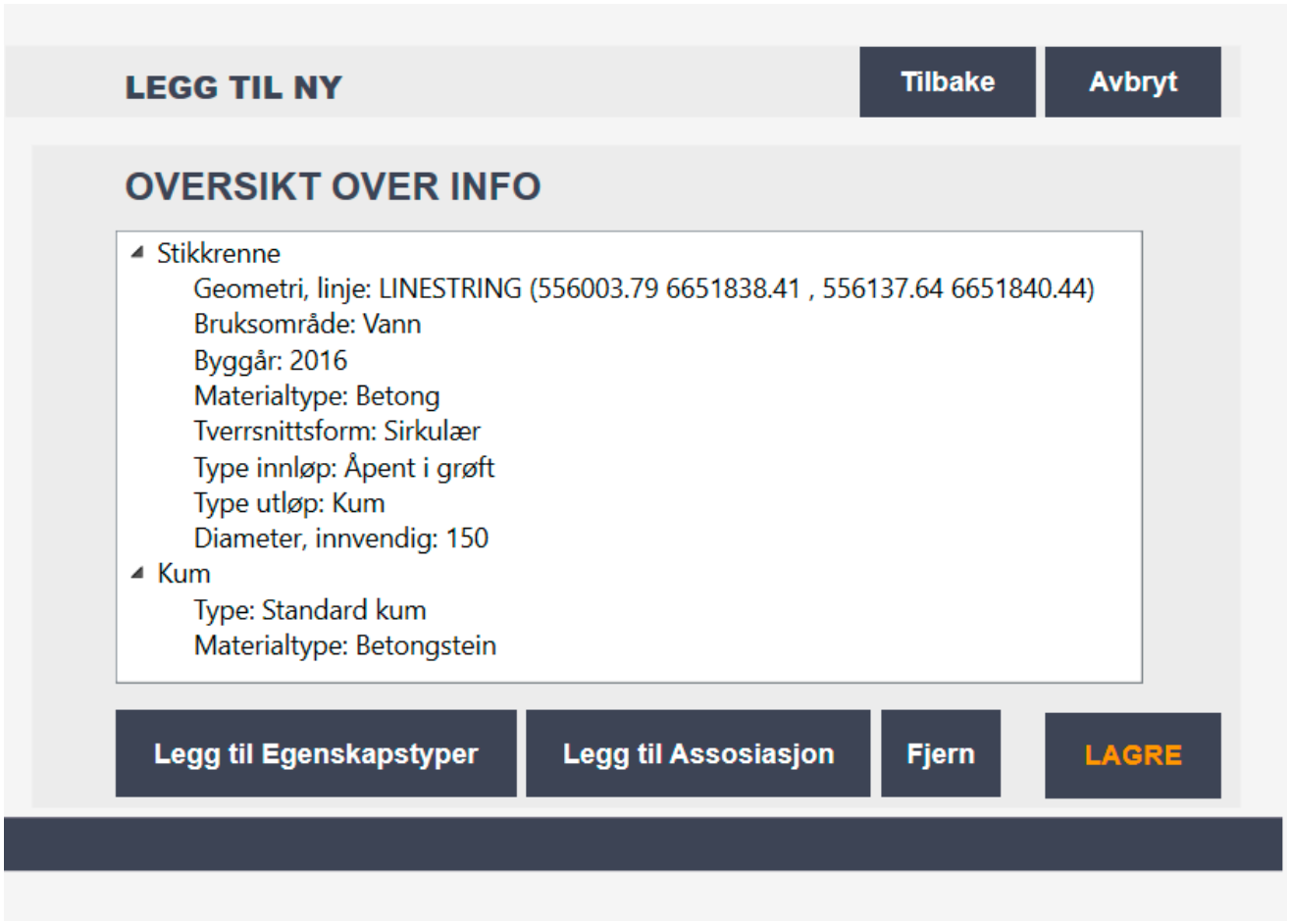
3 Når assosiert objekt og tilhørende egenskap er valgt, kan ønsket verdi settes eller velges i nedtrekksmenyen for egenskapsverdi.



4 5 Når egenskapsverdi er satt, kan + knappen benyttes dersom det skal legges inn flere objekter og/eller egenskaper. Når alt er lagt inn, velg **Legg til** for å komme tilbake til oversiktssiden.



Dersom det har blitt lagt inn informasjon om det assosierte vegobjektet kum, vil dette vises i oversiktslisten som vist i figuren under.

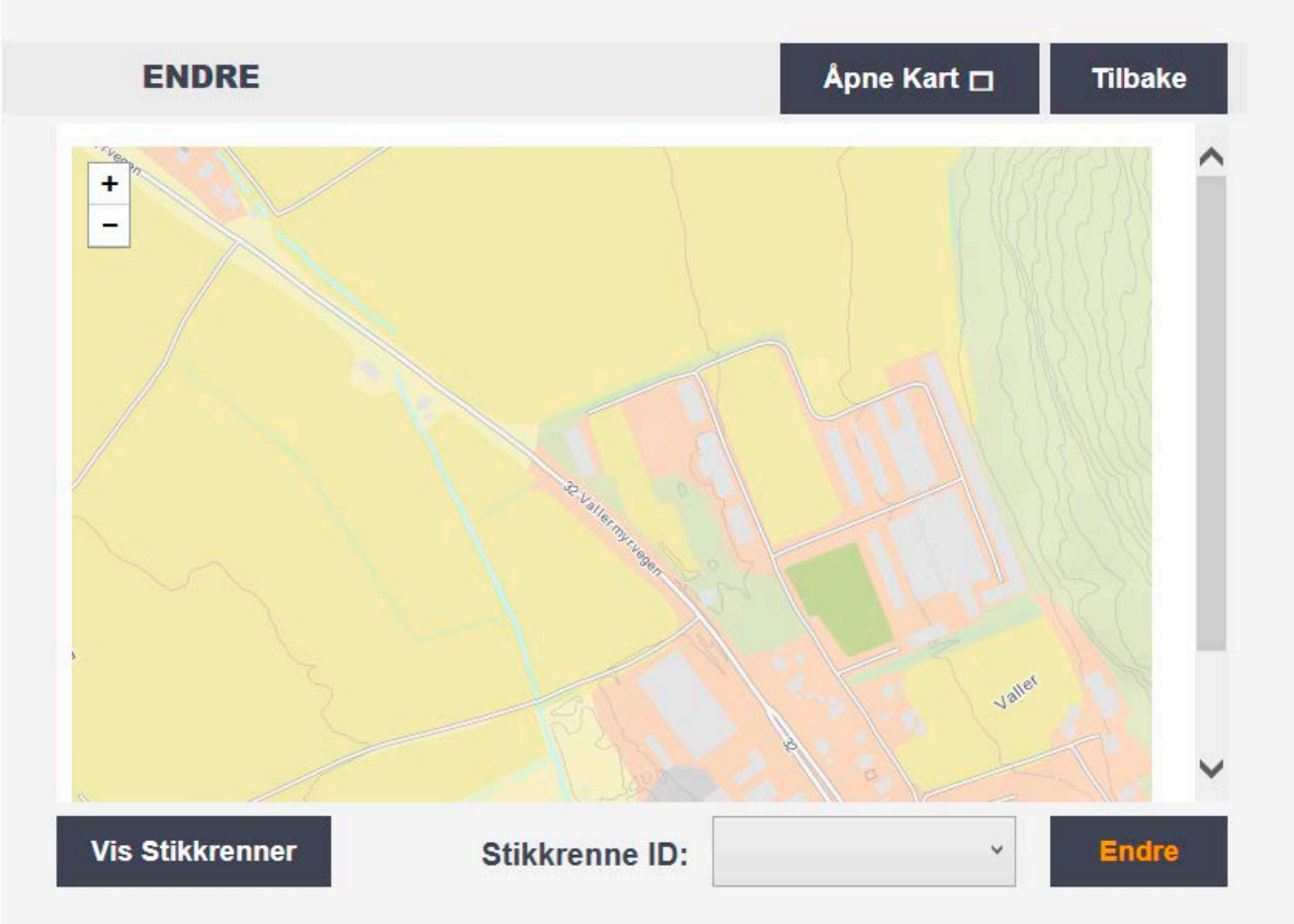


Når alt ser greit ut, velg **LAGRE.**

Endre

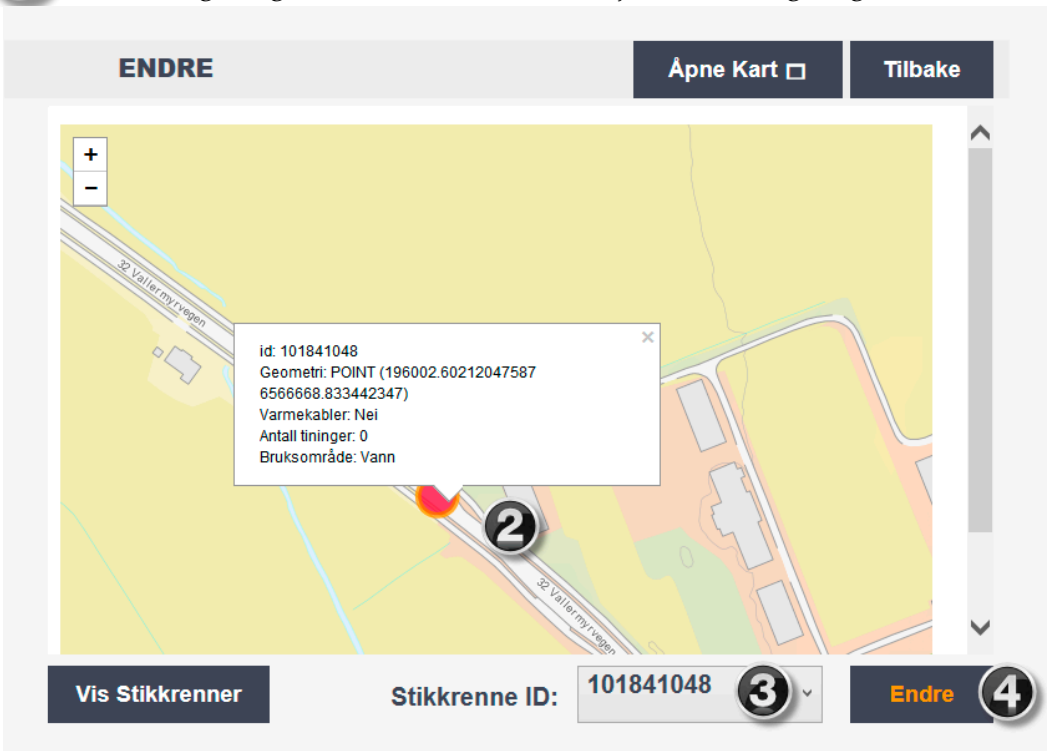
Figuren under viser knappen **Endre** fra startsiden.

Dersom **Endre** velges fra startskjermen, vises skjermbildet i figuren under. Denne siden gir mulighet for å vise stikkrenner i et geografisk område (maks 100 stikkrenner av gangen), velge stikkrenne ID i nedtrekksmenyen for så å velge endre.



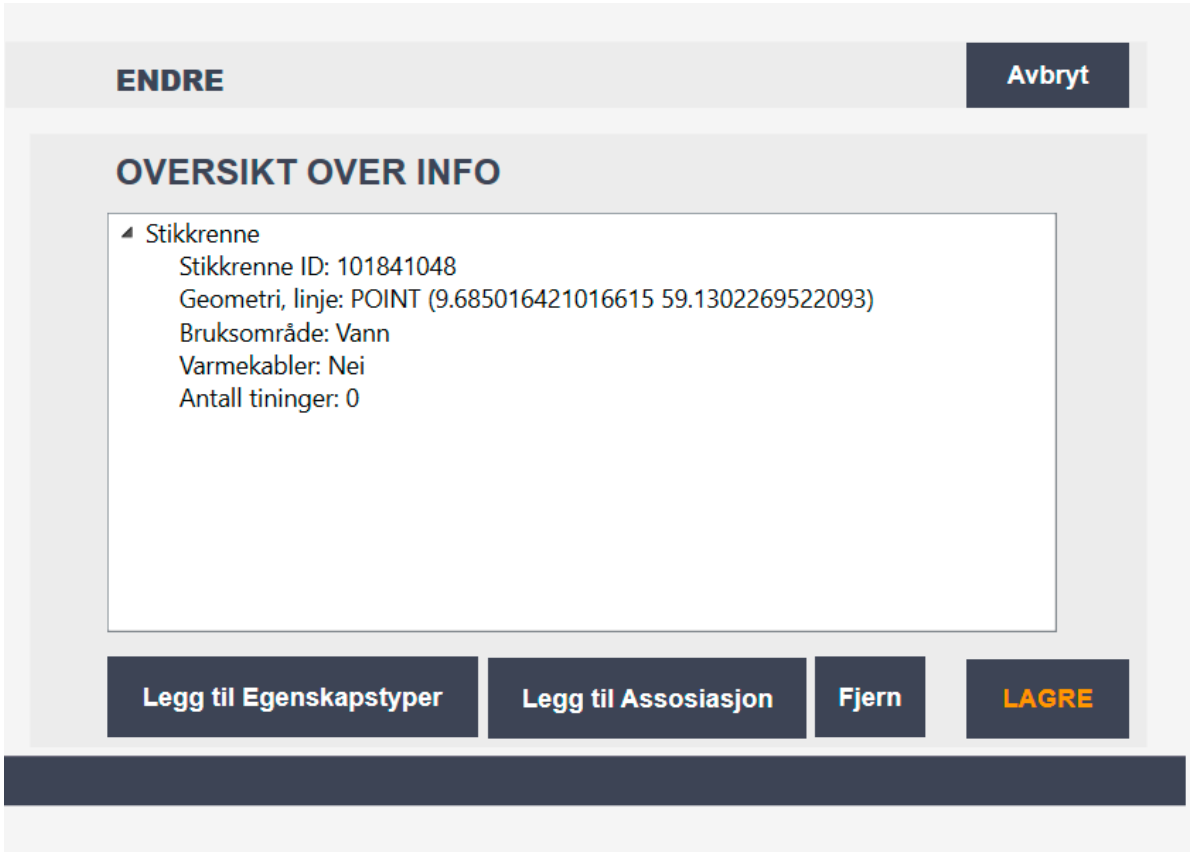
Trykk **Vis Stikkrenner** for å vise på kartet med oransje sirkler hvor stikkrennene innenfor et geografisk område befinner seg.

Klikk på sirkelen for å få informasjon om den aktuelle stikkrennen. For å gjøre endring, velg stikkrenne ID i kombinasjonsboksen og velg **Endre.**



Dersom bruker ønsker å maksimere vinduet, brukes **Åpne Kart.** Applikasjonsbildet blir da utvidet til skjermens størrelse og vil se ut som bildet vist nedenfor. Vinduet har samme funksjonalitet.

Informasjonen til den valgte stikkrennen vises i oversikt over info, som vist nedenfor. Når aktuelle endringer er gjort (som forklart tidligere i dokumentet), velg **Lagre.**



Appendix D - Test Cases

Test cases

Testing of software functionalities of the culvert registration tool.

Start Page

NR	Test Description	Comment
1	Does “x” close the entire program?	Yes
2	Does “Objektinfo” take you to the object information page?	Yes
3	Does “Legg til Ny” take you to the registration page?	Yes
4	Does the “Endre” button lead the view and edit culvert page?	Yes

Information Page

NR	Test Description	Comment
1	Is the culvert description appearing within the page?	Yes, if there is connection to the internet (stated requirement)
2	Does the cancel button take you back to the front page?	Yes

Registration Procedure

Page 1 – Geographical

NR	Test Description	Comment
1	Does the “Avbryt” button cancel the session and bring you back to the start page?	Yes
2	Is the map loading?	Yes, when connected to the internet (Stated requirement)
3	Is the blue marker appearing on the map?	Yes
4	Is the marker drag-able?	Yes

5	Is the start geometry stated in the according text field when clicking “Start”?	Yes
6	Is the end geometry stated in the text field when clicking “Slutt”?	Yes
7	Is road references retrieved from NRDB properly?	Yes, if there is no existing road reference at the marker location, a pop-up will notify the user
8	Does the text fields update if changing the position and clicking “Start”/”Slutt”?	Yes
9	Does the “Neste” button bring you to the first ‘add property’ page?	Yes

Page 2 – Add property page 1

NR	Test Description	Comment
1	Does the comboboxes fill with input property values from NRDB?	Yes, when connected to the internet (stated requirement)
2	Is it possible to write in the “Årstall” textfield? This field should be limited to integer values only.	Yes, but lack of limitation for upper value
3	Does pop-up boxes appear if the input of “Tverrsnitt” is either “Sirkulær”, “Rektanuglær” or “Ellipseform”?	Yes, all pop-up boxes appear
4	Is it possible to write in these textfields? These fields should be limited to integer values only.	Yes, but lacking upper limit for the input value
5	Can you click “OK”, without filling the text fields? A message should be given asking the user to fill before clicking “OK”.	When clicking “OK”, a message states that the text fields has to be filled to continue.
6	If the user don’t want to add these values, can the pop-up be cancelled by using “x”?	Yes, popup closes and type of shape is chosen
7	Does “Neste” take you to the next property input page?	Yes

Page 3 – Add property page 2

NR	Test Description	Comment
1	Does both comboboxes fill with input values from NRDB?	Yes, if connected to the internet (stated requirement)
2	Does “Tilbake” take you back to the previous page?	Yes
3	Does “Ferdig” take you to the overview?	Yes
4	Does “Neste” take you to the new property input page?	Yes

Page 4 – Add property page 3

NR	Test Description	Comment
1	Are all the buttons clickable?	Yes
2	Is only one of the two alternatives marked at a time?	Yes, one is coloured orange at a time
3	Does “Tilbake” take you back to the previous page?	Yes
4	Does “Ferdig” take you to the overview?	Yes
5	Does “Neste” take you to the new property input page?	Yes

Page 5 – Add property page 4

NR	Test Description	Comment
1	Does the comboboxes fill with input values from NRDB?	Yes if connected to the internet (stated requirement)
2	Does “Tilbake” take you back to the previous page?	Yes
3	Does “Neste” take you to the overview page?	Yes

Overview Page

NR	Test Description	Comment
1	Is the overviewbox filled with the values chosen in the registration process? If not, what is lacking?	Yes, all properties with values selected are displaying
2	Is it possible to click and mark a property in the list and remove it by using “Fjern”?	Yes, but if culvert and an associated object has the same property name, both will be removed
3	Does “Legg til Egenskaper” take you to the page for adding new properties?	Yes
4	Does “Legg til assosiasjon” take you to the page for adding associated objects?	Yes
5	Does “LAGRE” take you back to the start page?	Yes

Add new property

NR	Test Description	Comment
1	Is all the available properties showing in the list, starting at “PunktTilknyttning” and ending at “Høyde, passasje”?	Yes
2	When clicking the category buttons, do the list renew itself with according properties? P – “Bruksområde” – “Geometri, linje” B –” Har innløpsrist” - “Vedlikeholdsansvarlig” O – “Driftsmarkering” – “Geometri, flate” S – “Vinkel” – “Høyde, Passasje"	Yes
3	When choosing a property from the list, does the description under “beskrivelse” change to its according property description?	Yes
4	When a chosen property is marked, and if it has prefixed input values, are they appearing in the comobox?	Yes

5	When adding a value, is the text field adapted to its inputs? (Numeric, dates, etc.)	No, the fields are not adapted to data types of the selected property, nor limitations for entry.

Add associated object

NR	Test Description	Comment
1	Does the list box fill with the associated objects of the culvert?	Yes, if connected to the internet (stated requirement)
2	When choosing an associated object from the list, does the associated description show beneath the list box?	Yes
3	When choosing an associated object for the list, does the belonging properties show up in the listbox beneath “Tilhørende egenskaper?”	Yes
4	When choosing a property does the combobox fill with its prefixed input values (if any)?	Yes
5	Is the text field adapted for its type of input? (text, numeric, date etc.)	Not adapted to the specific type of property, lack of functionality for adding documentation such as pictures etc.
6	Does “Legg Til” take you to the overview, and is the new value added?	Yes
7	Does “Tilbake” take you back to the overview? Without any added culverts (if “+” haven’t been clicked)	Yes
8	The “+” button makes it possible to add more properties before going back to the overview. Add several properties and use “+” for each, when going back with “Legg Til” do they all appear in the overview?	Yes

Edit and Alter

NR	Test Description	Comment
1	Is the map loading?	Yes, when connected to the internet (stated requirement)
2	When clicking “Vis stikkrenner”, does orange and red dots appear?	Yes, if none exist there will be an error message
3	Zoom out, and re-click “Vis stikkrenner”, does more culverts show themselves (if there exist any)	Yes, if no exist there will be an error message
4	Does the combobox fill up with ID’s of the culverts displayed on the map?	Yes
5	When clicking a dot on the map, does a pop-up box appear with the according ID and properties?	Yes
6	When choosing an ID from in the combobox, and clicking “Endre” does it bring you to the overview?	Yes
7	When reaching the overview, is the same properties as seen in the pop-up box appearing (saying you chose this ID from the combobox)?	Yes
8	Does “LAGRE” take you to the start page?	Yes

Other

NR	Test Description	Comment
1	Is it possible to go back and forth between the property pages and the overview smoothly?	Yes
2	Is the previous chosen values showing in the comboboxes and text fields?	Yes
3	When cancelling a registration “Avbryt” at either the geographical pinning page or the overview, are all the previous variables removed?	Yes, all variables are wiped, except the headers “stikkrenne”, or associated object headers if they have been active prior

Appendix E - Summary Page

University College of Southeast Norway

MASTER'S THESIS, COURSE CODE FMH606

Student:	**Lilly Eirin Eikehaug**
Thesis title:	**Development of Prototype for Registration of Road Data**
Signature:	Lilly Eirin Eikehaug
Number of pages:	128
Keywords:	**C#, WPF, culvert, NRDB, NVDB, NPRA**

Supervisor:	Hans-Petter Halvorsen	Sign.:	Hans-Petter Halvorsen
Censor:		Sign.:	.....
External partner:	Statens Vegvesen	Sign.:	.....
Availability:	Open		

Archive approval (supervisor signature): Sign.: Hans-Petter Halvorsen **Date :** 2/6 - 2016

Abstract:

The Norwegian Public Roads Administration is responsible for the state and county public roads in Norway. This involves planning, construction and operation of the road network. NPRA manages the National Road Data Bank, which is a database that consist of information about the state, county and private roads.

When there is need for control or registration of new road objects, a registration application is used. However, the general registration applications are complex and usually handled by employees mainly working in the registration field. Based on this it is of interest to have a simplified application regarding a smaller field, such as road objects concerning water. The road object in focus for this thesis is the culvert. Culverts are installed across the roads and serves the purpose of leading water. They are crucial to handle large amount of rainfall along the road network.

The main goal of the thesis is to analyze, design and develop an application for road registration. The graphical user interface of the application is designed based on the need for a simplified application that could be used in fieldwork. Making a "tablet-friendly" application, with bigger fonts, buttons and fewer distractions.

The culvert registration tool retrieves road object information from NRDB by using the NRDB API, information such as culvert properties, existing culverts, road references etc. The registration procedure starts by presenting required and conditional properties. Later in the registration procedure, more properties of other categories can be added if needed, as well as associated objects of the culvert. The application uses an interactive map to set the geographical coordinates of the culvert and to display existing culverts for monitoring.

This thesis presents the research phase leading up to the development of the application and the documentation of the analyzing, design and development of the prototype for a registration application concerning culverts. A user manual for the application has been developed to show the program content and functionality in a systematic way.