YRKESHÖGSKOLAN
NOVIA

Automatisk dokumentation av Python-moduler

Matias Byggmästar

Examensarbete för ingenjör(YH)-examen

Utbildningsprogrammet för informationsteknik

Vasa 2017

EXAMENSARBETE

Författare:	Matias Byggmästar
Utbildning och ort:	Informationsteknik, Vasa
Handledare:	Kaj Wikman

Titel: *Automatisk dokumentation av Python-moduler*

Datum: 22.4.2017 Sidantal: 24

Abstrakt

Detta examensarbete gjordes på uppdrag av Neotide Ab, ett mjukvaruföretag i Vasa som specialiserar sig på databas- och nätverkslösningar för sjukvårdsdistrikt, kommuner och städer i Finland. Uppdraget var att skapa en tilläggsmodul till deras rapporteringsprogram Exreport. Tillägget skall automatiskt skapa dokumentation utgående från de databashämtningsmoduler som existerar i en kunds Exreport-system. Målet med dokumentationen är att ge kunder en översiktlig bild varifrån den data som rapporteras i Exreport har sitt ursprung.

Examensarbetet gjordes med programmeringsspråket Python och använder sig av dokumentationsverktyget Sphinx.

Resultat blev ett tillägg till Neotides rapporteringsverktyg som automatiskt generar en webbsida med all information om de olika modulerna och databaserna som finns i en specifik kunds rapporteringssystem.

Språk: svenska Nyckelord: Dokumentation, Python, Sphinx

OPINNÄYTETYÖ

Tekijä:	Matias Byggmästar
Koulutus ja paikkakunta:	Tietotekniikka, Vaasa
Ohjaaja:	Kaj Wikman

Nimike: Python-moduulien automattinen dokumentointi

Päivämäärä: 22.4.2017 Sivumäärä: 24

Tiivistelmä

Tämä opinnäytetyö tehtiin Neotide Oy:lle, joka on ohjelmistoyritys Vaasassa. Neotide on erikoistunut tietokanta- ja verkkopohjaisiin ratkaisuihin sairaanhoitopiireille, kunnille ja kaupungeille. Tehtävä oli kehittää lisäysmoduuli heidän raporttisovellukselle Exreport. Moduulin pitää automaattisesti luoda dokumentointi, joka perustuu tietokantamoduuleihin, jotka ovat olemassa asiakkaan Exreport-järjestelmässä. Dokumentoinnin tarkoitus on antaa asiakkaille yleiskatsauksen siitä, mistä Exreportissa oleva tieto on tullut.

Opinnäytetyö on tehty ohjelmointikielellä Python ja käyttää dokumentointityökalua Sphinx.

Tulos oli lisäys Neotiden raporttisovellukselle joka automaattisesti luo verkkosivun kaikilla tiedolla erilaisista moduuleista ja tietokannoista, jotka ovat asiakkaan raporttisovelluksessa.

Kieli: ruotsi Avainsanat: dokumentointi, Python, Sphinx

BACHELOR'S THESIS

Author:	Matias Byggmästar
Degree Programme:	Information Technology, Vasa
Supervisor:	Kaj Wikman

Title: Automatic Documentation of Python Modules

Date: April 22, 2017 Number of pages: 24

Abstract

This Bachelor's thesis was made for Neotide Oy, a software company in Vasa that specialises in Database- and network solutions for health care districts, municipalities and cities in Finland. The task was to create an addon module for their reporting system Exreport. The module shall automatically create documentation from the database modules that exists in a customer's Exreport-system. The purpose of the documentation is to give customers a summary of where the data that Exreport shows has its origin.

The thesis was made with the programming language Python and the documentation tool Sphinx.

The result is an addon to Neotides reporting system that automatically generates a website with information about the different modules and databases that exists in specific customers reporting system.

Language: Swedish Key words: Documentation, Python, Sphinx

Innehållsförteckning

1 Inledning

I detta kapitel presenteras uppdragsgivaren för detta examensarbete samt vad uppdraget gick ut på i korthet.

1.1 Uppdragsgivare

Oy Neotide Ab är ett programvaruföretag som är beläget i Vasa och har i skrivande stund 11 anställda. Företaget grundades år 1999 av Patrik Simons och Johan Kullas i Helsingfors men verksamheten flyttades till Vasa år 2001 (Neotide Ab, 2016).

Företaget är inriktat på nätverksapplikationer, databaslösningar och rapporteringsverktyg.

Neotide Ab:s kundgrupp består till största delen av olika sjukvårdsdistrikt, kommuner och städer i Finland. Några av företagets viktigaste produkter är:

- Exreport är ett webbaserat rapporteringsverktyg för sjukhus och kommuner. Programmet hämtar uppgifter från flera patient-, ekonomi- och personalsystem för att sedan sammanslå allt till olika rapporter för användare.

- SAI – Sjukhusets Antibiotika- och Infektionsuppföljningssystem. SAI är ett Windows-baserat system för uppföljning av infektioner på t.ex. en avdelning vid ett sjukhus.

- LogMonitor, ett system som är utvecklat för att övervaka patientsäkerheten genom att samla ihop logginformation från sjukhusets alla system som innehåller patient- eller personalinformation.

Förutom dessa produkter så har företaget även ett antal andra produkter för t.ex. hantering av måltider i sjukhus och ett system för hantering av patientjournaler.

1.2 Bakgrund och uppdrag

Exreport samlar dagligen in data från ett flertal olika källor genom att köra så kallade Batch-moduler. Batch-moduler är moduler i detta fall programmerade i Python som automatiskt hämtar data varje natt. Dessa moduler hämtar data från ett stort antal datakällor som kan vara av olika typer så som databaser, filer osv. Detta gör att det speciellt för kunder och deras IT-personal blir svårt att få en översiktlig bild varifrån all data kommer, när senaste körningen är gjord samt annan information om modulerna.

Därför blev uppdraget att skapa ett tillägg till Exreport som automatiskt samlar in information om de olika Batch-modulerna som existerar i en kunds system. Efter att all information har samlats ihop så sammanställs allting till en HTML-fil som länkas in i Exreport. Med hjälp av detta så får IT-personal hos en kund en översiktlig bild över var all data kommer ifrån samt annan tilläggsinformation om alla de olika modulerna.

2 Tekniker

I detta kapitel beskrivs programmeringsspråket Python, dokumentationsverktyget Sphinx, märkspråket ReStructuredText och Neotide Ab:s rapporteringsprogram Exreport samt andra tekniker som har använts i detta examensarbete.

2.1 Python

Python är ett programmeringsspråk med öppen källkod som utvecklades av Guido van Rossum under slutet av 1980-talet. Guiode van Rossum hade tidigare jobbat med ett programmeringsspråk kallat ABC och Python är till stor del inspirerat av ABC (Python Software Foundation, 2017)

År 1991 publicerade van Rossum källkoden för Python och släppte version 0.9. År 1994 släpptes version 1. Version 2 släpptes år 2000 och introducerade stor del av Pythons mest använda funktioner så som listomfattning och bättre minneshantering.

År 2008 släpptes version 3 (Python Software Foundation, 2017). Version 3 är en icke-bakåt kompatibel version dvs. att kod eller tillägg som är skrivet i Python 2.x inte nödvändigtvis fungerar med Python 3.x och vice versa. De största orsakerna till att göra version 3 icke-bakåt kompatibel var förändringen av kodningen av strängar så att alla

strängar sparas som ”Unicode” om inget annat anges. Språket städades även upp och förbättrades för att göra det lättare att lära sig och använda.

I dag så är Version 3 den rekommenderade versionen. Dock så finns det fortfarande ett betydande antal bibliotek som är skrivna i Python 2.x vilket leder till att det inte är möjligt att använda Version 3.x om ett projekt är beroende av ett sådant bibliotek. Den senaste släppta versionen är version 3.6 som släpptes i december 2016.

Python är skapat för att vara ett modernt programmeringsspråk med lättläst och enkel syntax. Det stöder flera olika typer av programmeringsmetoder t.ex. objektorienterad programmering och funktionell programmering. En del av Pythons grundfunktionaliteter och principer är:

- Dynamiska variabler. D.v.s. att vid deklaration av variabler behöver ingen speciell typ anges samt att det inte är variabeln som har en specifik typ utan det är själva värdet på variabeln som bestämmer typen.
- Använder indrag och blanksteg för att avsluta kodblock istället för tecken eller ord.
- Är ett tolkat programmeringsspråk och inte ett kompilerat programmeringsspråk.
- Automatisk minneshantering.
- Plattformsoberoende utveckling.

Exemplet nedanför är en enkel klass i Python. Klassen består av en konstruktor med en instansvariabel och ett par klassmetoder som kan addera eller subtrahera värdet på instansvariabeln.

Kodexempel 1, En enkel klass i programmeringsspråket Python.

```
class Number:
    def __init__(self, initial_number=0):
        self.number = initial_number
    def add(self, number):
        self.number += number
    def subtract(self, number):
        self.number -= number
my_number = Number(10)
my_number.add(5)
print(my_number.number)
```

2.2 Sphinx

Sphinx är ett dokumentationsverktyg utvecklat för Python som släpptes år 2008. Grundfunktionen med Sphinx är att konvertera märkspråket RestructuredText(ReST) till ett lättläst dokument så som HTML, LaTeX(PDF) med mera.

Sphinx genererar automatiskt en hierarkisk struktur i det valda utskriftsformatet av de ReST-filer som finns i ett projekt. Beroende på utskriftsformat så är det även möjligt att automatiskt generera innehållsförteckningar, sökfunktioner osv.

I dagsläget så är Sphinx det största dokumentationsverktyget för Python projekt och används bland annat för Pythons egna dokumentation (Python Software Foundation, 2017). Populariteten beror delvis på att det är lättanvänt och att det finns en stor mängd tilläggsmoduler som automatiserar användningen.

2.3 ReStructuredText

ReStructuredText är ett märkspråk baserat på StructuredText. StructuredText är ett liknande märkspråk som släpptes år 1996, det fanns dock vissa brister i det och därför valde Goodger att utveckla ReStructuredText som en förbättrad version av StructuredText (Goodger, 2013).

ReStructuredText släpptes år 2002 (Goodger, 2013) och är skapat för dokumentering av framförallt Python projekt men även för annan typ av dokumentation. Det är gjort för att vara lättläst och enkelt att konvertera till flera olika format.

Ett ReST-dokument är byggt upp med hjälp av en trädstruktur. Dokumentet är uppdelat i sektioner eller block av element som kan antingen tillhöra andra sektioner eller innehålla andra element. Elementen i sig själv är uppbyggda av tecken som representerar vad ett element har för betydelse. T.ex. kan ett element representera en titel, paragraf, lista eller tabell.

Figuren nedanför (Figur 1) är ett exempel på hur ett ReST-dokument är uppbyggt. Till vänster är dokumentets innehåll. Innehållet består av en huvudrubrik, två underrubriker, en kort lista och till sist en tabell. Till höger syns resultatet efter att dokumentet har konverterats till en webbsida med hjälp av Sphinx.

```
======
Rubrik
======
-----------
Underrubrik
-----------
*En lista*

- Punkt 1

- Punkt 2

- Punkt 3

-------------
Underrubrik 2
-------------
*En tabell*

+------------------------+------------+----------+----------+
| Rubrik                 | Rubrik 1   | Rubrik 2 | Rubrik 3 |
+========================+============+==========+==========+
| Rad 1                  | Rad 1      | Rad 1    | Rad 1    |
+------------------------+------------+----------+----------+
| Rad 1                  | Rad 1, kombinerad cell           |
+------------------------+------------+----------------------+
```

Rubrik

Underrubrik

En lista

- Punkt 1
- Punkt 2
- Punkt 3

Underrubrik 2

En tabell

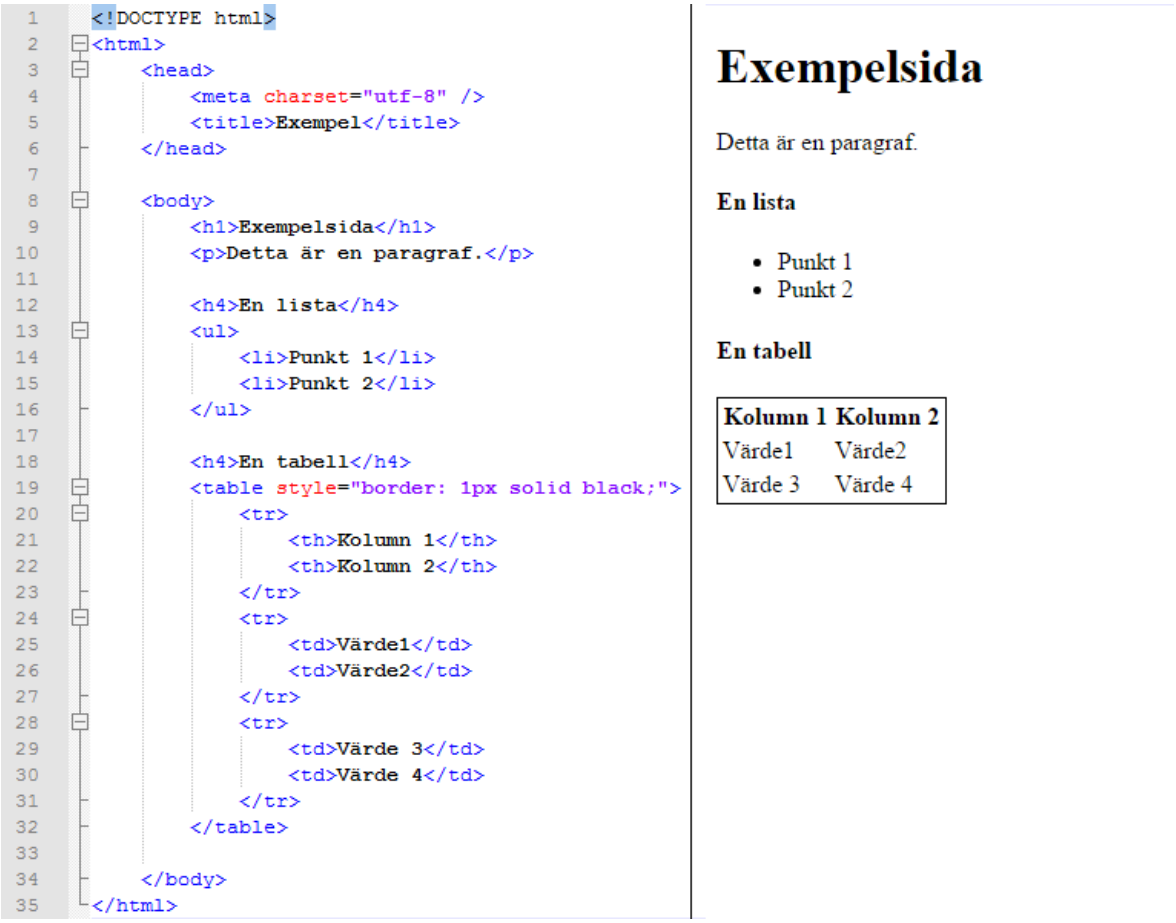
Rubrik	Rubrik 1	Rubrik 2	Rubrik 3
Rad 1	Rad 1	Rad 1	Rad 1
Rad 1	Rad 1, kombinerad cell		

Figur 1. Exempel på en genererad webbsida av dokumentationsverktyget Sphinx.

2.4 HTML

HTML eller HyperText Markup Language är ett märkspråk som används för att skapa webbsidor. Tillsammans med stilspråket CSS och skriptspråket JavaScript är dessa tekniker i dagsläget grunden som webbutvecklare använder för att bygga upp webbsidor (Flanagan, 2011).

Likt andra märkspråk så använder sig HTML av element för att beskriva hur ett dokument är uppbyggt och i HTML:s fall hur ett dokument skall tolkas av webbläsaren. Alla element är ordnande i en trädstruktur så att ett element kan innehålla flera andra element. Ett element beskriver sedan vad det innebär och hur det skall tolkas. T.ex. så kan ett element representera en titel, tabell eller paragraf så som det illustreras i Figur 2.



```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Exempel</title>
    </head>

    <body>
        <h1>Exempelsida</h1>
        <p>Detta är en paragraf.</p>

        <h4>En lista</h4>
        <ul>
            <li>Punkt 1</li>
            <li>Punkt 2</li>
        </ul>

        <h4>En tabell</h4>
        <table style="border: 1px solid black;">
            <tr>
                <th>Kolumn 1</th>
                <th>Kolumn 2</th>
            </tr>
            <tr>
                <td>Värde1</td>
                <td>Värde2</td>
            </tr>
            <tr>
                <td>Värde 3</td>
                <td>Värde 4</td>
            </tr>
        </table>

    </body>
</html>
```


Figur 2. Exempel på hur en webbsida är uppbyggd med hjälp av HTML.

2.5 Exreport

Exreport är Neotide AB:s rapporteringsprogram för patient-, ekonomi- och personalsystem. Exreport är ett webbaserat system som är skrivet i Python och använder Microsoft SQL Server som databassystem (Neotide Ab, 2015).

Alla kunder som använder Exreport har inte alltid liknande patient-, ekonomi eller personalsystem. Därför är programmet uppdelat i grupper eller så kallade ”siter” så att varje kund har sina egna specifika moduler för datahämtning och användargränssnitt. Men eftersom de flesta kunder är sjukhus och kommuner så är det ändå vanligt att det används samma system vid flera olika kunder. För dessa system så finns det grundläggande moduler som en specifik kunds moduler sedan kan köra eller ärva ifrån och lägga till de specialregler/tillägg som behövs.

Exreport är uppdelat i 3 större delar:

- **Datahämtning** genom de så kallade Batch-modulerna. I en Batch-modul specificeras varifrån man skall hämta data. T.ex. kan detta vara databaser, filer eller något annat lagringssätt. Oftast hämtas data från flera olika källor för att sedan kombineras i Exreports egna databas. En modul innehåller även information om t.ex. specialregler, filter samt sparar information om senaste körningar i en status tabell. Datahämtningar sker vanligtvis en gång per natt.

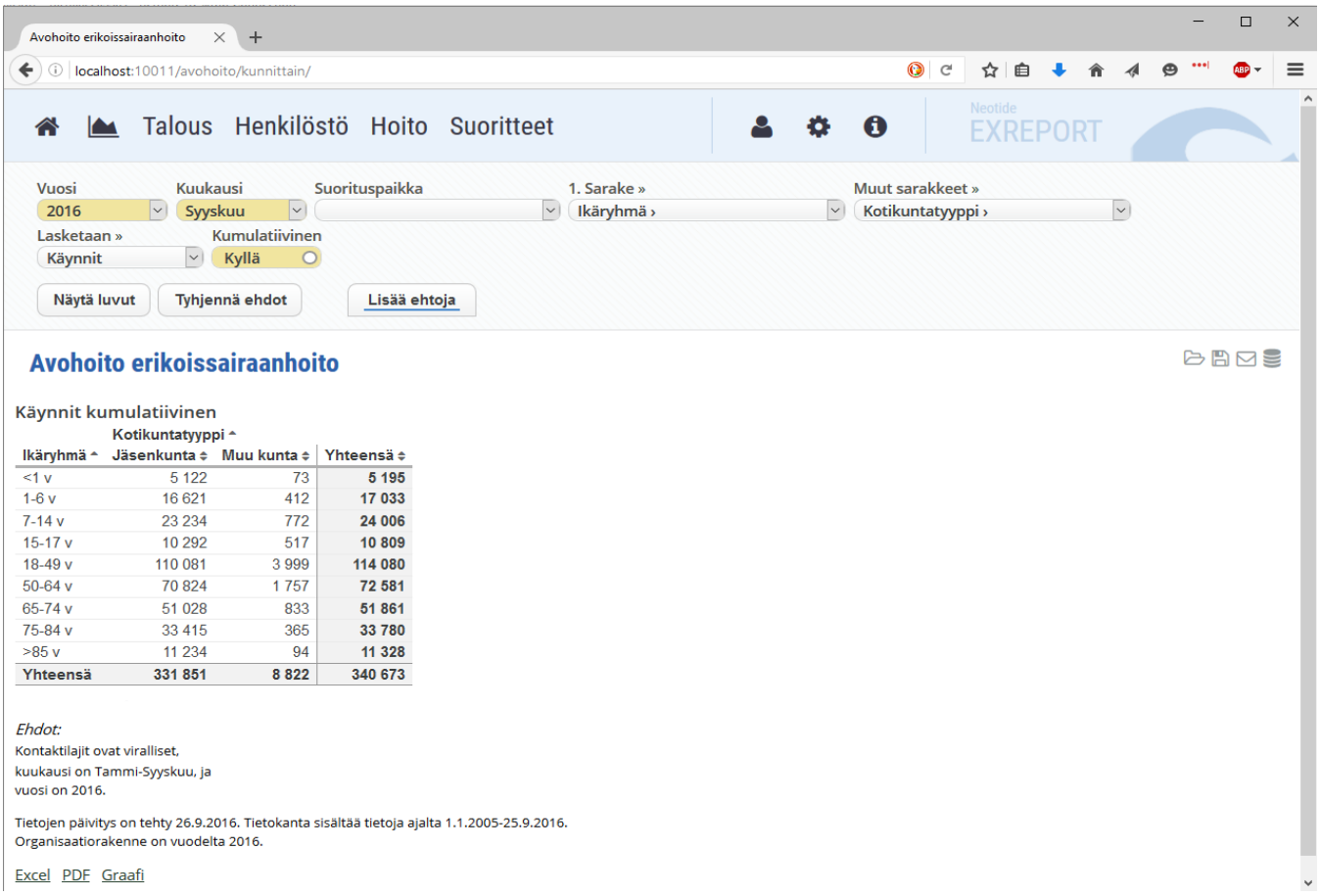
- **Rapportgenerator** är den del som är ansvarig för att skapa rapporter från det data som finns i Exreports egna databas. Rapportgeneratorn är i sig själv också uppdelad i tre större delar.

    - En del kallad **reportform** som bestämmer standardutseendet på en rapport och hur t.ex. de olika valmöjligheterna för begränsningar skall fungera.

    - En del kallad **renderreport** som genererar själva tabellen eller ett annat utskriftsformat från det data som väljs ut.

    - En del kallad **reportutil** hanterar hämtningen av data från tabeller och hur t.ex. tabeller skall kopplas ihop.

- **Användargränssnittet** är uppbyggt med hjälp av ett ramverk som heter Quixote. Quixote är ett webbaserat ramverk som är skapat för att vara flexibelt samt att ha bra prestanda. I Exreport så används en modifierad version av Quixote. Användargränssnittet för en rapport är baserat på att man har en tabell med data som kan begränsas eller utökas med olika valmöjligheter. En användare kan sedan klicka på cellerna i tabellen för att ta sig ner till enskilda fall-nivåer för att se mer detaljerad information. Figuren (Figur 3) nedanför är ett exempel på en rapport av öppenvårdsbesök i Exreport.



Figur 3. Exempel på en rapport i Exreport.

2.6 SQL och Microsoft SQL Server

SQL eller Structured Query Language är ett språk för att hantera data i en relationsdatabas. SQL utvecklades av IBM under 1970-talet (Chamberling & Boyce, 1974). Då kallades språket för SEQUEL och var utvecklat för att hantera data i IBM:s nyutvecklade relationsdatabas System R. Det blev snabbt det mest använda språket för databaser och år 1987 blev SQL klassificerat som en internationell standard av den internationella standardiseringsorganisationen (International Organization for Standardization, 2016)

För att hantera data i en relationsdatabas med SQL används olika kommandon och uttryck. Dessa kan delas upp i två större grupper:

- **Data Manipulation Language (DML)**-kommandon används för att hantera data i de objekt som existerar i en databas. Figuren nedanför visar ett exempel på ett vanligt DML-kommando. Kommandot hämtar alla rader ur en tabell med ett så kallat ”SELECT”-kommando.

```
SELECT * FROM ExampleTable;
```

Figur 4. Exempel på ett SQL-kommando för att hämta alla rader ur en tabell.

- **Data Definition Language (DDL)** uttryck används för att hantera strukturen på en databas och de objekt som existerar i databasen. Figuren nedanför visar ett exempel på ett vanligt DDL-kommando. Kommandot skapar en tabell i databasen med kommandot ”CREATE”

```
CREATE TABLE ExampleTable(
code CHAR(3) PRIMARY KEY,
text VARCHAR(30),
);
```

Figur 5. Exempel på ett SQL-kommando för att skapa en tabell.

Exreport använder Microsoft SQL Server som databashanterare. Microsoft SQL server är en databashanterare för relationsdatabaser som utvecklades under slutet av 1980-talet och version 1 släpptes år 1990. Den senaste släppta versionen i skrivande stund är SQL Server 2016 som släpptes 1 juni 2016.

Microsoft SQL server använder en dialekt av SQL som heter Transact-SQL(T-SQL). Dialekten är vidareutvecklingen av SQL och introducerar en del nya funktioner så som variabler och flödeskontroll (Byham & Guyer, 2017).

3 Utförande

I detta kapitel skrivs det om hur examensarbetet planerades, varför de tekniker som blev använda valdes samt hur arbetet utfördes.

3.1 Planering och grundläggande funktionalitet

Att skapa ett verktyg för att automatiskt generera dokumentation för kunder är något som Neotide har diskuterat internt tidigare men som inte har genomförts. Därför fanns det redan en del idéer och förslag om hur det skulle genomföras.

Det första som diskuterades var om det skulle göras som ett helt separat program eller om det skulle skapas som ett tillägg till Exreport. Till slut föll valet på att det skulle bli ett tillägg till Exreport. De största orsakerna till att göra det som ett tillägg istället för ett separat program var att det finns en stor mängd funktioner och moduler i Exreport som kan återanvändas t.ex. hämtningen av data från databasen och tillgång till de moduler som körs.

Den andra större punkten som diskuterades var i vilket format som den slutgiltiga dokumentationen skulle sparas i. De valmöjligheter som diskuterades var att antingen spara det som ett textdokument eller som en webbsida. Valet blev att dokumentationen skulle sparas som en webbsida på grund av användarvänlighet samt att det är enklare att implementera. Men detta val är inget som är slutgiltigt eftersom Sphinx kan generera dokumentation i ett flertal olika format (Brandl, 2017). Därför går det enkelt att anpassa efter kundens behov om de begär efter dokumentationen i ett annat format.

3.2 Krav och mer specifik funktionalitet

Efter att den grundläggande funktionaliteten var färdigdiskuterad så började en diskussion om de mer specifika kraven och vad dokumentationen faktiskt skall innehålla. De slutgiltiga kraven över vad som skall finnas med blev:

- Vilken typ av överföring det är och vilket format är källan i. Dvs. hämtas data från en tabell, fil eller från någon annan källa.
- Tidpunkt för när den senaste körningen har blivit utförd.
- I vilken tabell/tabeller i Exreports databas sparas den data som hämtats.
- Information om källdatabasen. Så som användarnamn, serveradress, portar osv.
- Information/beskrivning av enskilda moduler. De flesta moduler innehåller en variabel med en kort beskrivning.

Det fanns även ett annat viktigt krav och det är att hela tillägget skulle köras som enskild Batch-modul. Detta betyder att i praktiken så tillhör största delen av koden till en klass som heter Batch som sedan ärver från en annan grundklass.

Denna grundklass innehåller en del standardvariabler och funktioner som de ärvda klasserna sedan antingen använder eller skriver över. De flesta av dessa funktioner och variabler är för att enklare kunna hämta och hantera rader som man hämtar från databasen. I detta examensarbete så används inte största delen av dessa variabler och funktioner utan det är cirka fyra till fem av dem som används. Bland de viktigaste och mest använda av dem är variabeln som anger vilken konfigurationsfil som används eftersom i den finns all information om källdatabasen, server osv. En av de andra viktiga variablerna är namnet på tabellen i Exreports databas var största delen av data sparas.

3.3 Val av teknik

I detta kapitel skrivs det om motivationen bakom att använda de valda teknikerna och hur de valdes.

3.3.1 Python

Valet av programmeringsspråk var från början uppenbart eftersom Python är det programmeringsspråk som Neotide använder sig av till de flesta projekt. Samt att det är det programmeringsspråk som Exreport är skrivet i. Därför blev det självklart att även detta examensarbete skulle göras huvudsakligen i Python eftersom det är ett tillägg till Exreport.

Om examensarbetet skulle blivit ett separat program istället för ett tillägg så skulle det självklart varit möjligt att skriva det i ett annat modernt programmeringsspråk men även då så är sannolikheten stor att valet hade blivit att programmera det i Python.

3.3.2 Sphinx

Att välja dokumentationsverktyg var mer komplicerat än att välja programmeringsspråk. Det fanns en del alternativ som beaktades. Det första som diskuterades var om ett dokumentationsverktyg överhuvudtaget skulle användas eller om tillägget i sig själv skulle skapa t.ex. en webbsida/textfil. Valet blev dock att använda ett dokumentationsverktyg eftersom det automatiskt har stöd för t.ex. sökfunktioner och teman.

När det väl var bestämt att ett dokumentationsverktyg skulle användas så blev valet enklare. Neotide har tidigare använt Sphinx för dokumentation i tidigare projekt. Sphinx är också det officiella dokumentationsverktyget för Python och det finns en stor mängd resurser så som guider, tillägg och annan information som kan användas vid behov.

3.3.3 HTML

Som tidigare nämndes så blev det även diskuterat hur dokumentation skulle sparas. Det fanns en del alternativ och bland dessa alternativ så var det huvudsakligen att spara dokumentation som ett textdokument eller som en webbsida som var de mest relevanta alternativen även fast Sphinx kan generera andra filformat.

Valet blev dock att spara det som en webbsida eftersom det är det vanligaste formatet som används när man genererar dokumentation med hjälp av Sphinx. Det är även det enklaste att använda samt att man får en del funktionalitet automatiskt (sökfunktion, utseende osv).

3.4 Genomsökning av moduler

Varje kunds system består av ett flertal moduler som automatiskt körs varje natt. En lista med sökvägar samt olika optioner till dessa moduler sparas i en skild fil som beskriver vilka moduler som skall köras samt de optionerna om hur de skall köras. Bland annat så kan olika optioner vara att moduler är inaktiverade eller att de är beroende av andra moduler.

Om man importerar listan i denna fil så är det möjligt att dynamiskt importera modulerna för att sedan hämta ut den information som man är intresserad av från dessa moduler.

Allting genomförs med hjälp av en funktion som returnerar en lista över de moduler som har importerats med hjälp av sökvägarna. I början av funktionen definieras en ny lista över ignorerade moduler som inte skall importeras. Exempel på ignorerade moduler är t.ex. moduler för underhåll av databasen, uppdatering av cache osv.

De moduler som inte blir ignorerade importeras med hjälp av en modul som heter importlib. Denna modul är en del av Python och kan användas direkt genom att importera den. Modulen innehåller funktioner och metoder för att enklare importera och hantera andra moduler. I detta examensarbete så var det en specifik funktion som var intressant och användes.

Denna funktion heter ”import_module” och är en enklare version av Python’s grundläggande import-funktion. Med hjälp av ”import_module” så importeras varje modul från listan över sökvägar och de läggs till i en ny lista som den huvudsakliga funktionen returnerar.

3.5 Generering av RestructuredText-filer

Generering av RestructuredText filerna är uppdelad i två större delar. Den första delen består av själva skapandet och skrivningen av filen som sedan skall konverteras till HTML. Även denna del är sedan uppdelad i två delar.

3.5.1 Skrivning av RestructuredText-filen

Till att börja med så hanteras alla databaser och all information om dem. Detta utförs genom att importera en konfigurationsfil som innehåller all information i form av ”dictionaries”. Tillägget går sedan igenom varje ”dictionary” som finns i konfigurationsfilen och skriver ut dess nyckel och värde i en lista med hjälp av en annan klass (Se kapitel 3.5.2). Värden så som lösenord till databaserna filtreras naturligtvis bort. Figur 6 visar ett exempel på hur den information om en databas som finns lagrad i

konfigurationsfilen ser ut i dokumentationen.

Figur 6. Exempel på hur information om en databas ser ut i dokumentation jämfört med kod.

Den andra delen av den första delen består av utskriften av information om själva Batch-modulerna. Det hela börjar med en loop över alla databaser som sorteras i alfabetisk ordning. Detta för att varje modul som använder en specifik databas skall grupperas per databas i dokumentationen. Efter det så körs en till loop över alla de moduler som tidigare har importerats. Loopen innehåller till först en del kontroller för att se vilka variabler som

modulen innehåller. Till exempel så behöver inte en modul innehålla en variabel för en databas om all data i modulen hämtas från filer.

Sedan hämtas en del av informationen från en tabell i Exreports databas. I databasen finns en tabell med information om t.ex. när en körning på just denna modul senast har blivit gjort, tidigaste data och senaste data.

För att hämta ut informationen från tabellen används en funktion som tar namnet samt vilken kolumn man är intresserad av som parametrar. I detta fall så hämtas tidpunkten när en modul senast har blivit kört.

All annan information finns sparade som variabler i själva modulerna och eftersom de har blivit importerade så kan man enkelt komma åt denna information och skriva ut den med hjälp av en annan klass (se kapitel 3.5.2).

För att undvika att stänga och öppna filen flera gånger så lagras all utskrift först i en variabel för att sedan skrivas ut med en förbättrad version av Pythons ”open”-funktion från ett annat bibliotek vid namn ”codecs”. Motivationen till att använda ”codecs.open” istället för den vanliga funktionen för att öppna filer är att ”codecs.open” tar emot extra parametrar så som kodning(encoding).

3.5.2 Hjälpklass för RestructuredText-utskrift.

Eftersom RestructuredText är ett märkspråk så kräver det ett visst format på utskriften för att det skall fungera. Istället för att då vara tvungen att i varje utskrift ta i beaktande syntaxen för varje element i ReStructuredText-format så blev det ett val mellan att hitta en lämplig modul som redan existerar eller skapa en egen klass.

De moduler som redan existerade var dock för stora, föråldrade eller inte tillräckligt omfattande. På grund av dessa orsaker och att det egentligen inte rör sig om en stor mängd funktioner så blev valet att skapa en egen klass.

Den klass som skapades är en enkel klass med funktioner som omger utskrift/text med de korrekta elementen.

Kodexempel 2, Exempel på en del av hjälpklassen för utskrift av RestructuredText.

```
class RstOutput:
    @staticmethod
    def title(title, type):
    """ Return a ReStructuredText title of the specified type."""
     return type * len(title) + '\n' + title + '\n' + type *len(title)
    @staticmethod
    def list_title(title, type):
    """ Return a ReStructuredText header for a key-value list."""
     return type + header + type + ' \n\n'
```

Klassen innehåller förutom de föregående exemplen (Kodexempel 2) även andra funktioner för de element som används mest så som:

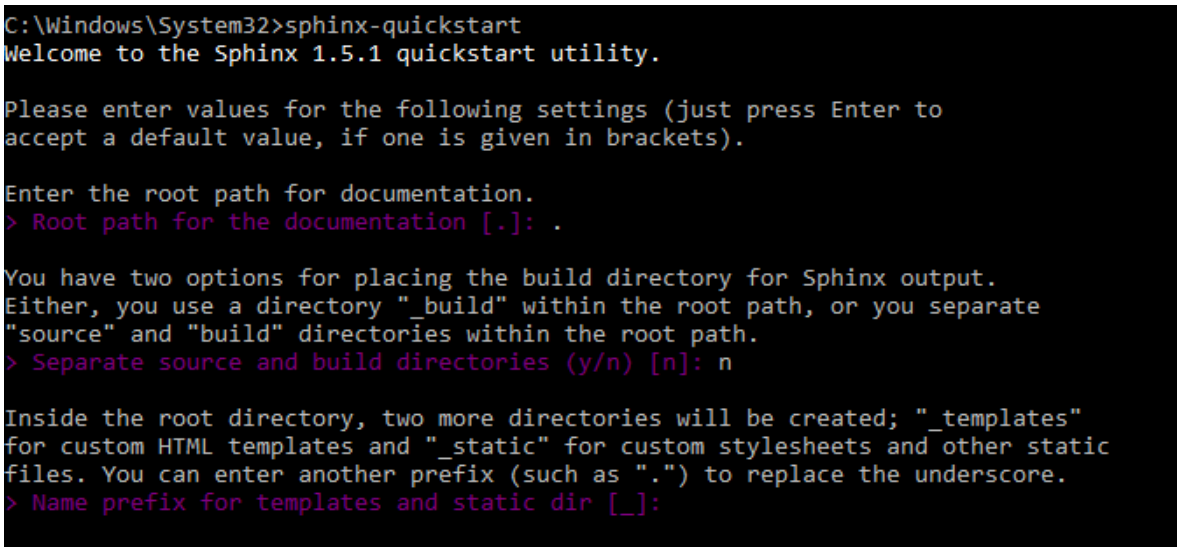
- Rubriker och underrubriker som omges med specifika tecken som är av samma längd som texten.
- Listor och nästlade listor påbörjas med ett specifikt tecken och skall alltid avslutas med två radbrytningar.

Alla funktioner tar emot en variabel med den information som skall skrivas ut. En del av funktionerna tar dessutom emot en extra parameter som säger vilken typ av t.ex. titel det skall vara. Dessa extra parametrar är nödvändiga eftersom det finns flera olika sätt att skapa titlar, rubriker osv. i ReStructuredText.

3.6 Installering och konfiguration av Sphinx

Det finns ett flertal olika sätt att installera Sphinx men i detta fall så installerades Sphinx med hjälp av kommandotolken och Pythons egna pakethanterare som heter pip. Efter att Sphinx har installerats så kan man skapa ett dokumentationsprojekt med hjälp av att använda kommandot ”sphinx-quickstart” i en kommandotolk.

När man kör detta kommando så är man tvungen att svara på ett par frågor och fylla i värden så att Sphinx kan skapa projektet med de rätta inställningarna. Nedanför i Figur 7 visas ett par av de frågor som ställs när man skapar ett nytt projekt.



Figur 7. Exempel på körning av kommandot "sphinx-quickstart".

Efter att detta kommando har blivit kört så skapar Sphinx en skild mapp som innehåller ett par standardfiler med olika värden beroende på de svar som gavs tidigare. Denna mapp innehåller ännu ingen dokumentation utan det skapas i ett senare skede.

I de flesta fall så genererar man dokumentation från ReStructuredText filer med hjälp av en skriptfil som Sphinx har skapat men detta kräver att man har en konfigurationsfil som är inställd med korrekta värden. Denna konfigurationsfil måste innehålla t.ex. namnet på projektet och vissa rubriker som i detta fall skulle vara namnet på t.ex. ett sjukhus.

Men eftersom tillägget är skapat för att fungera på flera kunders system utan ändringar så används inte denna skriptfil utan istället används Sphinx API(applikationsprogrammeringsgränssnitt).

I denna API finns en funktion med namnet ”sphinx_build”. Denna funktionen har en speciell egenskap i form av att det är möjligt att generera dokumentation utan att ha en skild konfigurationsfil. Funktionen kan istället ta emot de värden som finns i konfigurationsfilen i form av en lista som argument. Därför är det möjligt att generera dokumentation nästan helt oberoende av kund genom att skicka med i listan namnet på kunden och andra kundspecifika värden. Det finns även funktionalitet i Exreport för att automatiskt hämta ut dessa värden så att de inte manuellt behöver deklareras.

Kodexempel 3, Exempel på användning av funktionen ”sphinx_build”.

```
def run_sphinx(self):
    sphinx.build_main(['sphinx-build',
                       '-C',
                       '-D', 'language=fi',
                       '-D', 'html_logo=images/logo.png' %
                       '-D', 'project=%s' % config.site.upper(),
                       '-D', 'copyright=2017, Neotide',
                       '-D', 'master_doc=index',
                       self.doc_path,
                       os.path.join(self.doc_path, 'build')
                       ])
```

Exemplet ovan kör funktionen ”sphinx_build” med en lista över argument. Det första argumenten ”-C” ställer in så att ingen konfigurationsfil skall användas. Istället ställs en del av de värden som finns i en konfigurationsfil in med hjälp av ”-D” och sedan värdet. De sista två argumenten är sökvägarna till ReStructuredText-filerna samt var den genererade dokumentation skall sparas.

3.7 Användargränssnitt

Användargränssnittet består av två enklare sidor och är uppbyggt med hjälp av Sphinx standardtema som kallas Alabaster.

Utseendemässigt så är det ett väldigt enkelt och stilrent tema. Temat stöder både Python 2 och 3 och är av så kallad responsiv design. Responsiv design är en designmetod för att anpassa webbsidor till olika skärmstorlekar. Metoden har ökat i popularitet de senaste åren allt eftersom antalet enheter med olika skärmstorlekar har ökat drastiskt.

Man kan även skräddarsy temat genom en konfigurationsfil men allt som blev ändrat i detta examensarbete var att t.ex. logon blev bytta till Neotides logo samt andra mindre förändringar så som storleken på marginaler osv.

Funktionsmässigt så består webbsidan som tidigare nämndes av två delar.

Den första delen är en introduktionssida som innehåller länkar till olika delar av dokumentationen samt en sökfunktion. Det är en standardsida som Sphinx automatiskt skapar och innehållet bestäms av en ReStructuredText fil som heter ”index.rst”. I denna fil bestäms vilka länkar som skall t.ex. synas i menyn osv. Nedanför i Figur 8 visas ett exempel på hur en startsida kan se ut.

Figur 8. Exempel på hur en startsida ser ut.

Den andra delen är själva innehållet av dokumentationen. Sidan består av en snabbmeny till vänster som innehåller länkar till de olika delarna av dokumentation samt en sökfunktion. Själva innehållet är uppdelat i två delar.

Den första delen är listan över de olika datakällorna som existerar i en kunds Exreport-system. Dessa är ordnade alfabetiskt och innehåller då den information som tidigare hämtades.

Den andra delen är listan över de olika modulerna som existerar. Dessa är först grupperade och ordnande per datakälla. Sedan kommer namnet på modulen och information om den modulen. I Figur 8 visas ett exempel på hur en modul dokumenteras.

Figur 9. Exempel på hur information om en modul ser ut i dokumentationen.

4 Resultat

Resultatet blev ett tillägg till Exreport som automatiskt skapar webbaserad dokumentation för en kunds IT-personal. Med hjälp av tillägget så kan IT-personalen hos en kund enkelt kontrollera t.ex. vilka system som är i användning, varifrån all data till de rapporter som är i användning kommer ifrån samt när data senast har blivit uppdaterad osv.

Tidigare när kunder ha frågat efter liknande dokumentation så har man manuellt gått igenom de olika modulerna och fyllt i textdokument som skickats till kunder. Nu kan detta undvikas och dokumentation hålls även uppdaterad och blir inte utdaterad.

För tillfället används detta tillägg vid ett sjukvårdsdistrikt men eftersom det blev skapat för att vara anpassningsbart så är det en relativt enkel process att installera det för flera kunders Exreport-system om behovet av det skulle uppstå.

5 Diskussion

Överlag så är jag nöjd med resultatet av tillägget. Att skapa dokumentation är inte någon favoritsysselsättning för de flesta programmerare men nu kan man undvika en stor del av det manuella arbetet.

I framtiden kommer tillägget att vidareutvecklas genom att lägga till mer information så som alla specifika tabeller om en modul hämtar information från flera tabeller, vilka kolumner, flera tidsstämplar osv. Webbsidan kommer även att utseendemässigt uppdateras för att påminna mer om Exreport samt att allt eftersom det läggs till mer information till dokumentationen så kommer den delas upp i flera sidor. För tillfället innehåller inte index-sidan så mycket information eftersom det bara finns en sida som den länkar till. Men i framtiden då tillägget vidareutvecklas och det har lagts till fler sidor så blir index-sidan mer relevant och nödvändig.

Den andra saken som kunde göras är att installera tillägget för flera kunder. Just nu så är det installerat för ett sjukvårdsdistrikt. Eftersom tillägget inte innehåller någon kod som är specifikt för just en kund så är det enkelt att installera tillägget för andra kunders Exreport system.

Den tredje större saken som kunde vidareutvecklas är så att tillägget även genererar dokumentation för rapporter och inte bara för Batch-moduler. Detta är något som skulle

behövas eftersom även för rapporter så skapas all dokumentation manuellt. Tyvärr så innebär det en stor mängd arbete för att vidareutveckla tillägget så att det även skapar dokumentation om det eftersom rapporter är svårare att generera/hämta information om.

Examensarbetet har även varit lärorikt och intressant att utföra. Även fast jag tidigare har erfarenhet av Python så var det flera okända bibliotek och moduler som jag fick lära mig och använda.

De andra större lärdomarna var hur Sphinx används och fungerar. Jag har inte tidigare använt Sphinx så det var intressant att lära sig hur det används och eftersom det är det mest populära dokumentationsverktyget för Pythons samt att Neotide tidigare har använt det för dokumentation så tror jag att det även var en väldigt bra erfarenhet för framtiden att få använda och lära mig det.

Även RestructuredText var helt nytt för mig och det var lärorikt att få lära sig hur det fungerar ihop med t.ex. Sphinx.

Källförteckning

Brandl, G., 2017. *Sphinx Documentation.* [Online]
Tillgänglig vid: https://media.readthedocs.org/pdf/sphinx/stable/sphinx.pdf
[Använd 12 1 2017].

Byham, R. & Guyer, C., 2017. *Transact-SQL Reference,* u.o.: u.n.

Chamberling, D. D. & Boyce, R. F., 1974. *IBM Research.* [Online]
Tillgänglig vid: http://www.almaden.ibm.com/cs/people/chamberlin/sequel-1974.pdf
[Använd 5 5 2017].

Flanagan, D., 2011. *JavaScript - The definitive guide.* 6:a red. u.o.:O'Reilly Media.

Goodger, D., 2013. *An Introduction to reStructuredText.* [Online]
Tillgänglig vid: https://tools.ietf.org/doc/docutils-doc/docs/ref/rst/introduction.html
[Använd 12 1 2017].

International Organization for Standardization, 2016. *ISO/IEC 9075-1:2016,* u.o.: International Organization for Standardization.

Neotide Ab, 2015. *Exreport presentation*

Neotide Ab, 2016. *Företaget.* [Online]
Tillgänglig vid: http://neotide.fi/foretaget.html
[Använd 12 1 2017].

Python Software Foundation, 2017. *About these documents.* [Online]
Tillgänglig vid: https://docs.python.org/3/about.html
[Använd 12 1 2017].

Python Software Foundation, 2017. *Releases.* [Online]
Tillgänglig vid: https://www.python.org/download/releases/3.0/
[Använd 12 1 2017].

Python Software Foundation, 2017. *Why was Python created in the first place.* [Online]
Tillgänglig vid: https://docs.python.org/3/faq/general.html#why-was-python-created-in-the-first-place
[Använd 12 1 2017].

Figurförteckning

Förteckning över kodexempel