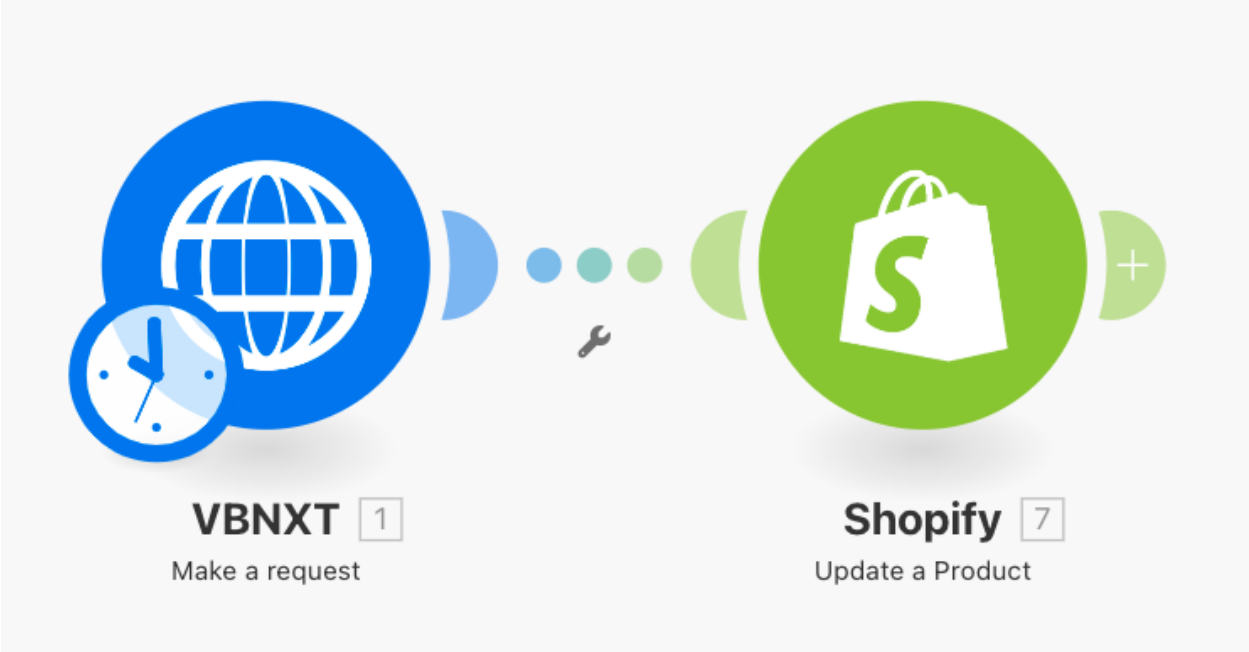



# Integration mellan Visma Business NXT och Shopify

Examensarbete inom högskoleprogrammet Datateknik (TIDAL)

TOBIAS ADRIAN, ELINOR DAHLQVIST

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK, CSE

CHALMERS TEKNISKA HÖGSKOLA
Göteborg 2025
www.chalmers.se

Examensarbete 2025

# Integration mellan Visma Business NXT och Shopify

TOBIAS ADRIAN, ELINOR DAHLQVIST


CHALMERS

Institutionen för data- och informationsteknik, CSE
Chalmers Tekniska Högskola
Göteborg 2025

Integration mellan Visma Business NXT och Shopify
TOBIAS ADRIAN, ELINOR DAHLQVIST



Handledare: Johannes Åman Pohjola, Chalmers
Handledare: Thomas Adrian, Pector
Examinator: Nick Smallbone, Chalmers

Examensarbete 2025
Institutionen för data- och informationsteknik, CSE
Chalmers Tekniska Högskola
SE-412 96 Göteborg
Telefon +46 31 772 1000

Omslagsbild: Ett teoretiskt scenario bestående av två moduler i Make.com, för att visualisera integrationen.

Skriven i LaTeX
Göteborg 2025

Integration mellan Visma Business NXT och Shopify
TOBIAS ADRIAN, ELINOR DAHLQVIST
Institutionen för data- och informationsteknik, CSE
Chalmers Tekniska Högskola

# Sammanfattning

I detta projekt utvecklades en integration mellan affärssystemet Visma Business NXT och e-handelsplattformen Shopify. Projektets mål var att automatisera överföringen av produkter, kunder, lager och priser från Visma Business NXT till Shopify, samt ordrar i motsatt riktning. Projektet utfördes hos företaget Pector AB, som hjälper kunder att integrera e-handelsplattformar mot Vismas plattformar för lager.

Integrationen byggdes på plattformen Make.com med hjälp av API-baserad datahantering. Under projektets gång undersöktes också om .NET skulle kunna användas som plattform istället för Make och ifall .NET skulle kunna leda till en välfungerande integration. En avgränsning med projektet var att endast testdata skulle användas och inte ”riktig” data i en webbshop.

Slutresultatet av projektet blev en fungerande integration som uppfyllde kraven för projektet och lade en grund för vidareutveckling.

Nyckelord: Integration, API, databas, webbshop, GraphQL, Make.com, VBNXT, Shopify, .NET.

# Innehåll

**Figurer** **ix**

**1 Inledning** **1**
1.1 Bakgrund . . . 1
1.2 Syfte . . . 1
1.3 Mål . . . 2
1.4 Avgränsningar . . . 2

**2 Teknisk Bakgrund** **3**
2.1 Visma Business NXT . . . 3
2.2 Shopify . . . 3
2.3 Make.com . . . 3
2.3.1 Användning av Make.com . . . 4
2.4 GraphQL . . . 6
2.5 .NET . . . 8

**3 Metod** **9**
3.1 Arbetsprocess . . . 9
3.2 Krav . . . 9
3.2.1 Funktionella krav . . . 9
3.2.2 Icke-funktionella krav . . . 9
3.3 Arkitektur och design . . . 10
3.4 Utvärdering . . . 10

**4 Genomförande** **11**
4.1 Introduktion till projektet . . . 11
4.1.1 Förberedelser . . . 11
4.1.2 Test av andra metoder . . . 12
4.2 Integrationen . . . 12
4.2.1 Relevanta moduler . . . 13
4.2.1.1 NXT Token och NXT Query . . . 13
4.2.1.2 Stored Variables . . . 13
4.2.1.3 Webhook . . . 13
4.2.2 Scenarier . . . 14
4.2.2.1 Produkter . . . 14
4.2.2.2 Priser . . . 15
4.2.2.3 Kunder . . . 16

4.2.2.4 Lager . . . . . 16
4.2.2.5 Ordrar . . . . . 17

**5 Resultat** **19**

**6 Slutsats** **21**

**Bibliography** **23**

# Figurer


2.1 Make-scenario med två moduler, en webhook och en modul för Google Kalkylark. Scenariot triggas av att en order skapas i Shopify. . . . . 4
2.2 Webhook med output efter att en order har skapats i Shopify. . . . . 5
2.3 Delar av output från ordern visat i ett Google Kalkylark. . . . . 5
2.4 Förslag på typ av modul för att skapa en ny modul. Varje modul är grupperad under en ”app”, beroende på vilken tjänst som ska integreras. 6
2.5 Fråga i GraphQL för att få varje karaktärs namn. Frågan är döpt till ”Chars”. Namnen hämtas ur fältet ”results” i fältet ”characters”. . . . 6
2.6 JSON-objekt med namn, som returneras från frågan i Figur 2.5 . . . 7
2.7 GraphQL-fråga för att få namn och status, samt JSON-objektet som returneras till höger. . . . . 7

4.1 Samtliga scenarion i Make som används i integrationen . . . . . 12
4.2 De sex första modulerna som finns i scenarierna för produkter, priser, kunder och lager. . . . . 14
4.3 Make-scenario för att uppdatera information om produkter i Shopify. 15
4.4 Make-scenario för att uppdatera produkters priser i Shopify. . . . . 15
4.5 Make-scenario för att uppdatera information om kunder i Shopify. . . 16
4.6 Make-scenario för att uppdatera lager på produkterna i Shopify . . . 17
4.7 Make-scenario för att skicka ordrar från Shopify till VBNXT . . . . . 18

# 1

# Inledning

## 1.1 Bakgrund

E-handel är en stor del av dagens samhälle och ses som en naturlig del av handeln. Det har därför blivit allt viktigare att kunna lagra data från webbsidor med produkter, ordrar och allt runtomkring på ett effektivt sätt. Detta kan ofta innebära att skapa integrationer mellan gamla databaser mot nya system som kan lagra data på ett annat sätt. Målet med detta projekt var att skapa en integration mellan en databas och en e-handelsplattform.

Projektet genomfördes hos Pector AB. Pector är ett konsultföretag som hjälper andra företag att utveckla sina affärssystem. En av Pectors kunder skulle börja använda Shopify istället för en annan webbshop och ville behålla den information som fanns i Visma Business NXTs (hädanefter VBNXTs) databas. VBNXT är en molnbaserad tjänst för att lagra all data för en webbshop samt lagerhantering. Kunden ville samtidigt att det som fanns i VBNXT skulle synkroniseras med Shopify. Shopify är en e-handelsplattform som gör mer än en webbshop. Shopify hanterar både butiksgränssnitt och tillhandahåller verktyg för API-integrationer.

All information kring produkter, kunder, priser och lager finns i VBNXT. Informationen kommer att synkroniseras med Shopify med hjälp av integrationen. Integrationen är viktig för att den möjliggör en molnbaserad lagring samt fler automatiserade tjänster. Kunden använde sig tidigare av Visma Business, som installeras lokalt på servrar medan VBNXT istället lagras i molnet [1].

## 1.2 Syfte

Syftet med projektet är att skapa en integration och undersöka om Make är ett relevant verktyg att använda. Behovet av integrationen uppkom vid ett uppdrag med en av företagets kunder som ville integrera sina varor, ordrar och lager i sin Shopify-affär. En bit in i projektet valde kunden att istället önska en integration med Shopify-konkurrenten WooCommerce. Vid det laget bestämde sig företaget att Shopify-uppdraget skulle fortsätta ändå, för att ge framtida kunder en möjlighet att använda integrationen.

## 1.3 Mål

Målet med projektet var att skapa en API-baserad integration mellan VBNXT och Shopify. Planen var att använda scenarion i Make för de olika funktionaliteterna och integrera produkter, kunder, priser och lager. Under projektets gång undersöktes även andra alternativ, som exempelvis att använda .NET för att skapa integrationen.

Målet var att VBNXT ska få tillgång till ordrar tidigare än vad som sker idag. VBNXT får information från ordrar i en XML-fil efter att ordern har skickats till lagret och betalningen har genomförts. Önskemålet var att VBNXT skulle få information innan orderinformationen skickats till lagret.

VBNXT nuläge och mål, med pilar som visar informationsflödet:

- Nuläge: Webbshop → Lager → webbshop, betalning Klarna → VBNXT får i efterhand in en XML-fil med ordern
- Målet: Webbshop (Shopify) → VBNXT → Lagret (exempelvis spårning) → VBNXT, levererad → pinga till Shopify (betalning)

## 1.4 Avgränsningar

Integrationen ska integrera data från VBNXT till Shopify och inte åt andra hållet. Projektet använde endast det nya VBNXT och inte Visma Business.

Projektet ska skapa en prototyp av en integration. Detta innebar att endast testdata användes när integrationen utvecklades, för att undvika eventuella problem och buggar som kan skapas om ”riktig” data används.

Under projektets gång och efter diskussioner med Pector lades ett krav till om att ett scenario för ordrar skulle skapas. Det innebär att ordrar som skapas i Shopify ska integreras till VBNXT vilket från början inte var inom projektets avgränsade område.

# 2

# Teknisk Bakgrund

## 2.1 Visma Business NXT

Visma Business NXT är ett affärssystem för att hantera lagerinformation, produkter, redovisning och liknande information. Detta krävs inom ett företag för att möjliggöra en effektiv hantering av lager och logistik. VBNXT har utvecklats för att främst hjälpa medelstora samt stora företag att hantera sina affärsprocesser på ett effektivt, iterativt och automatiserat vis. Till skillnad från tidigare versioner av Visma Business systemet så är VBNXT molnbaserat [2]. Detta gör att företag som använder sig av VBNXT inte skulle behöva någon serverinfrastruktur och deras medarbetare skulle ha bättre tillgång till affärssystemets tjänster.

Den viktigaste aspekten av VBNXT, i förhållande till det här projektet, är dess kraftfulla stöd för integrationer. Systemet har ett öppet API som möjliggör sammankoppling med andra tjänster, vilket gör att systemet kan anpassas utifrån företags behov och önskemål.

## 2.2 Shopify

Shopify är en e-handelsplattform som hanterar majoriteten av det som behövs för att bedriva e-handel. Shopify fungerar å ena sidan som en webbshop där kunder kan köpa produkter och å andra sidan sköts även allt annat kring betalning, lager och frakt av Shopify. Shopify har goda förutsättningar för att integreras mot system för lagerhantering, fraktlösningar samt bokföring [3].

Shopify möjliggör testning av webbsidor genom att ha funktioner för att starta en egen test-webbsida med produkter och kunder. Då kan den som skapar webbsidan använda alla funktioner för administratörer, detta för att testköra integrationen utan att behöva använda sidan med riktiga produkter och kunder.

## 2.3 Make.com

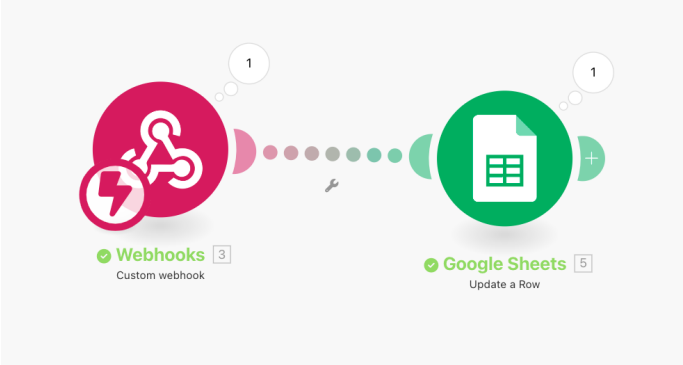
Make är en plattform som är till för att automatisera system och processer genom att koppla ihop API:er och databaser [4]. Make är ett visuellt verktyg som består av scenarion och moduler. Ett *scenario* är ett arbetsflöde som körs sekventiellt. Varje scenario är uppbyggt av *moduler* vilka är specifika byggblock som utför en specifik

uppgift. Den specifika uppgiften kan vara att skicka en fråga för att extrahera information från en databas, en webhook, för att upptäcka när en ny order skapas i en butik eller lägga in data i en webbshop.

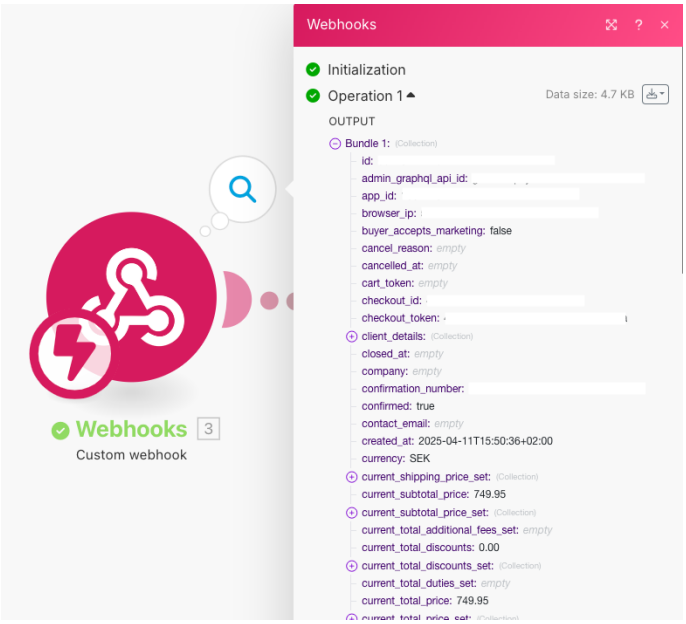
### 2.3.1 Användning av Make.com

Ett exempel på hur Make kan användas är att skapa en webhook som triggas när en ny order skapas i Shopify. En webhook är en modul som triggas av en händelse. Händelsen kan vara i det här fallet att en order skapas i Shopify. Informationen som behövs från ordern läggs sedan in i ett Google Kalkylark. Detta för att spara informationen från den nya ordern. Detta används inte i projektets integration då det inte är effektivt att lagra information om en order på detta sätt eftersom, det är svårt att iterera över rader i kalkylarket via Make. Exemplet som visas i denna delen är mest för att visa hur ett enkelt scenario ser ut och fungerar.

För att göra detta behövs ett nytt scenario skapas. En översikt av scenariot ses i Figur 2.1. Scenariot består av två moduler, en för webhooken och en för att uppdatera kalkylarket. Webhooken kopplas till Shopify så att modulen triggas när en ny order skapas. Informationen som skickas från Shopify till webhooken visas i Figur 2.2. Informationen skrivs sedan in i kalkylarket, Figur 2.3. Blixten på webhooken innebär att det är den modulen som körs först när scenariot triggas.




**Figur 2.1:** Make-scenario med två moduler, en webhook och en modul för Google Kalkylark. Scenariot triggas av att en order skapas i Shopify.

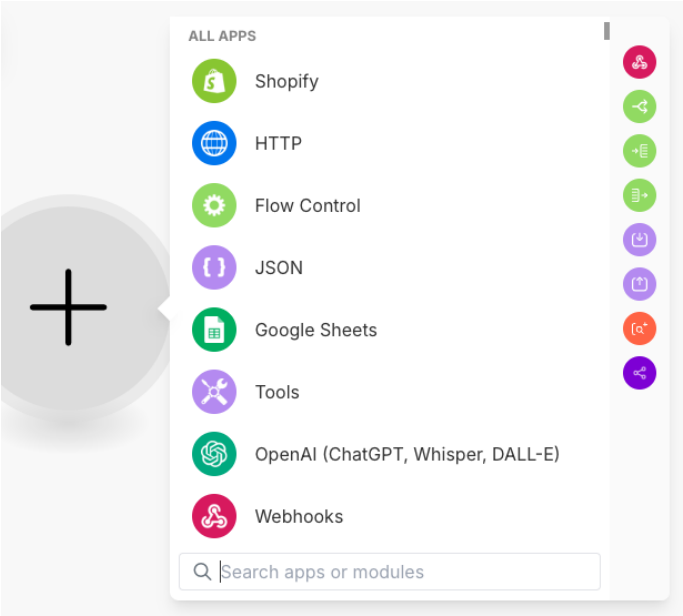



**Figur 2.2:** Webhook med output efter att en order har skapats i Shopify.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | id | confirmation number | created at | current price | currency |
| 2 | | | 2025-04-11T15:50:36+02:00 | 750.35.00 | SEK |
| 3 | | | | | |

**Figur 2.3:** Delar av output från ordern visat i ett Google Kalkylark.

Det finns många olika typer av moduler att välja på i Make, se exempel i Figur 2.4. Detta innebär att det går att integrera mellan flertalet system och det möjliggör en mängd olika integrationer. Ett annat exempel på moduler är att Make har verktyg för att hämta GraphQL-frågor inkluderade i JSON-strängar, frågorna används för att hämta data från API:er samt databaser. Moduler kan också vara parsers, moduler för iterering eller moduler för att skicka GraphQL-frågor.




**Figur 2.4:** Förslag på typ av modul för att skapa en ny modul. Varje modul är grupperad under en ”app”, beroende på vilken tjänst som ska integreras.

## 2.4 GraphQL

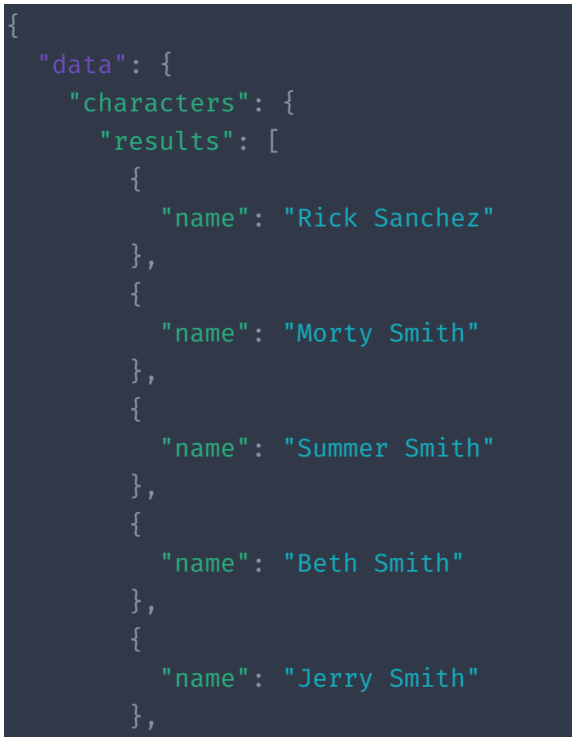
GraphQL är ett frågespråk skapat av Facebook [5]. Språket tillåter användaren att specificera vilken data som ska hämtas, muteras eller övervakas. Detta gör att språket kan ses som mer modernt och flexibelt till skillnad från de mer traditionella REST-API:erna.

En fråga i GraphQL kan se ut som i Figur 2.5.

```
query Chars {
  characters{
  results{
  name
  }}
}
```

**Figur 2.5:** Fråga i GraphQL för att få varje karaktärs namn. Frågan är döpt till ”Chars”. Namnen hämtas ur fältet ”results” i fältet ”characters”.

I detta fall frågas API:et efter alla namn på de karaktärer som finns i databasen. En fråga ställs till en databas med information om Rick and Morty serien[1]. Det som returneras är ett JSON-objekt, vilket visas i Figur 2.6.




**Figur 2.6:** JSON-objekt med namn, som returneras från frågan i Figur 2.5

GraphQL gör det enkelt att fråga efter specifik data och bara få ut det som är relevant. I Figur 2.7 är även ”status” inkluderad i resultatet, som då, i detta fall, visar om karaktären är vid liv eller inte.


**Figur 2.7:** GraphQL-fråga för att få namn och status, samt JSON-objektet som returneras till höger.

GraphQL-frågespråket är nödvändigt för att kunna använda VBNXTs API. Utan kunskap om GraphQL blir det avsevärt svårare att hantera mängden information som ska överföras mellan VBNXT och Shopify. GraphQL möjliggör därför en integration av all nödvändig data.

[1]https://rickandmortyapi.com/graphql

## 2.5 .NET

.NET är en plattform utvecklad av Microsoft. Plattformen används för att utveckla applikationer och är känd för sitt starka stöd för objektorienterad programmering och höga prestanda [6]. I detta arbetet har utvecklingen i .NET skett med C# och Azure Functions. Dessa i kombination skulle då vara ett alternativ till de funktioner som Make redan erbjuder för det här projektet, dock till ett betydligt lägre pris.

# 3

# Metod

Detta kapitel tar upp projektets övergripande process, kraven som ställdes upp i början av projektet, designen av integrationen samt en utvärdering av hur ett lyckat resultat förväntas se ut.

## 3.1 Arbetsprocess

I projektet användes en iterativ utvecklingsprocess för att kunna få kontinuerlig feedback. Detta gjorde att det var enklare att upptäcka fel i processen och snabbt kunna byta riktning om något inte fungerade.

## 3.2 Krav

Projektet är en del av en betydligt större integration mellan Shopify och VBNXT. Kraven har därför utgått från samtal med Pector om deras önskemål, då kundens krav skulle vara för omfattande för den här projektstorleken.

### 3.2.1 Funktionella krav

1. Integrationen skall överföra produkter, lager, priser och kunder från VBNXT till Shopify.

2. Ett krav på att ordrar skall överföras från Shopify tillbaka till VBNXT lades till senare under projektet.

### 3.2.2 Icke-funktionella krav

1. Integrationen ska förstärkas av en beskrivande dokumentation. Dokumentationen ska innehålla information om projektet, vägledning för andra utvecklare samt hur arbetet ska föras vidare.

2. Integrationen måste byggas med avseende på iteration och vidare utveckling, eftersom detta projekt enbart utgör en liten del av helheten.

## 3.3 Arkitektur och design

Integrationen görs helt och hållet i plattformen Make. Integrationsstrukturen är utformad för enkel iterering samt begriplighet. Scenarion i Make är uppdelade med avseende på låg koppling och hög kohesion, ett etablerat tillvägagångssätt som gör integrationen enklare att förstå. Varje scenario ansvarar för sin egen del i programmet och de beror inte på varandras funktionalitet.

För att programmet ska fungera så behöver vissa variabler sättas i Make. Dessa variabler är kundspecifika, vilket innebär att de ändras för att fungera med kundens företag. Variablerna, som används i flera scenarion, kan därför sättas samtidigt med hjälp av Makes och VBNXTs respektive inbyggda variabellagringsfunktion. Variablerna förklaras mer utförligt längre ner i under-underavsnitt 4.2.1.2.

Ytterligare dokumentation har skapats under projektets gång, där varje del beskrivs utförligt. Dokumentationen innehåller för- och nackdelar med integrationen, samt alternativa vägar som utvecklare kan ta.

## 3.4 Utvärdering

Ett lyckat resultat av projektet skulle vara en omfattande integration som uppfyller alla krav från avsnitt 3.2 samt en integration som har möjlighet att fortsätta vidareutvecklas. Detta sker enbart då alla kraven för projektet är uppfyllda. De funktionella kraven är mätbara vilket gör att det är möjligt att se om de har uppfyllts.

Kraven gav en tydlig struktur för arbetet och gjorde det enklare att förstå varje steg i processen. Användandet av Make gjorde också arbetet tydligare på grund av dess användarvänlighet. Mycket av den tid som skulle ha lagts på kod kunde istället räknas bort och användas på andra områden av integrationen och projektet.

# 4

# Genomförande

I detta avsnitt ges en närmare inblick i hur projektet utfördes. Främst kommer delar av den färdiga integrationen visas, samt bilder och figurer för att göra det tydligare. Det kommer också redovisas försök till andra metoder som inte användes i den slutgiltiga integrationen. Detta för att kunna ställa de olika metoderna mot varandra i slutsatsdelen.

Målet med projektet, som tidigare nämnts i avsnitt 1.3, är att designa och implementera en integration mellan VBNXT och Shopify. Företag som använder VBNXT ska kunna automatiskt överföra information och data direkt till Shopify.

## 4.1 Introduktion till projektet

Projektets första skede bestod av ett möte tillsammans med kunden som önskade integrationen. Mötet hölls med de två handledarna till projektet samt två anställda på kundföretaget. Mötet bestod av en introduktion till projektet och redovisning av kundens önskemål.

I början av projektet var det främst diskussioner kring hur och med vilken plattform arbetet skulle utföras. Tillsammans med handledarna beslutades det om att testa den relativt nya plattformen Make.com. Pector testade fortfarande Make och det var inte bestämt om hur eller när det skulle användas på bästa sätt när kraven bestämdes. Alternativet till Make skulle vara utveckling med .NET. Detta skulle däremot vara mer tidskrävande då den största fördelen med Make är hur lätt det är att använda. Make valdes därför som huvudplattform medan .NET behölls som reservalternativ.

### 4.1.1 Förberedelser

Förberedelserna av projektet bestod främst av att skapa konton samt att lära sig använda VBNXTs hemsida. Pector har tillgång till flera demobolag på VBNXT. Ett demobolag innehåller endast testdata, för att utvecklare i säkerhet ska kunna utveckla, utan att riskera att av misstag dela med sig av känslig information. Även fast testdata användes så behövde hemliga nycklar hanteras som exempelvis API-nycklar.

GraphQL, som beskrivits i avsnitt 2.4, är sättet integrationen skickar frågor till

VBNXTs API. VBNXT tillhandahåller en ”GraphiQL”-hemsida där det går att ställa frågor direkt. GraphiQL är en integrerad utvecklingsmiljö för GraphQL-frågor, i detta fallet en GraphiQL-webbsida specifikt för VBNXT.

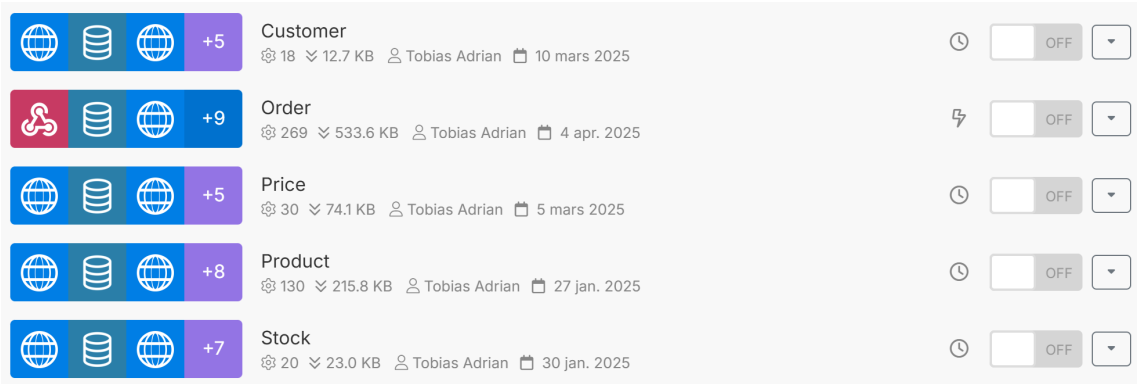
### 4.1.2 Test av andra metoder

Under projektets gång undersöktes olika metoder för att genomföra integrationen. I början av projektet användes Make då det rekommenderades av handledaren. Efter en månad med Make så bestämdes det att integrationen istället skulle utföras med .NET-utveckling. Projektet bytte plattform eftersom handledaren på Pector var tvungen att byta från Make till .NET i ett annat projekt på grund av höga kostnader. Beslutet togs eftersom det i Make finns en risk att operationerna ökar exponentiellt och därmed också priset. Detta innebär att .NET har en signifikant lägre kostnad än Make.

Klienten som användes i C# och som skapades för VBNXT hade vissa begränsningar vilket innebar att utförandet av integrationen blev mer komplext än planerat. En klient med en större funktionalitet än den som fanns tillhandahållen behövdes. Biblioteket i .NET visade sig bristfällig i några delar av implementationen, en bristfällighet som Make inte visade på. Med tanke på situationen beslutades det att integrationen skulle återvända till att utföras i Make. Beräkningar utfördes på kostnaden och risken för en exponentiellt ökande kostnad bedömdes vara tillräckligt låg för att kunna återvända till Make.

## 4.2 Integrationen

Projektet resulterade i en integration byggd helt på plattformen Make.com. I Make skapades fem scenarion: Customer, Order, Price, Product och Stock, se Figur 4.1. Uppdelningen gjordes för att kunna dela upp integrationen i delar som vid projektets början verkade rimliga. Scenarion som delade upp de olika data som skulle skickas bedömdes då mest rimligt för att få en struktur för arbetet.




**Figur 4.1:** Samtliga scenarion i Make som används i integrationen

Integrationen kunde då föra över all relevant data kring kunder, pris, produkter och lager från Visma Business NXT till Shopify och möjliggöra en webbshop i Shopify.

När en order skapas i Shopify så läggs ordern till i VBNXTs databas och lagerstatus uppdateras.

### 4.2.1 Relevanta moduler

Scenarion i Make byggs upp av ett flertal moduler. Varje modul har olika funktionalitet och används i olika delar av programmet. Några av dessa används i alla scenarion i integrationen.

#### 4.2.1.1 NXT Token och NXT Query

Den första modulen i nästan alla scenarion är av typen ”Make a Http Request”, som benämnts ”NXT Token”. Denna modul används, i våra fall, för att hämta API-nyckeln som behövs för att nå VBNXTs API. Den kombineras oftast med en modul, som vi har döpts till ”NXT Query”, som då använder sig av API-nyckeln för att ställa en fråga till VBNXT. I detta fall har GraphQL-data använts, men eftersom Make-moduler bara tar in data i json format har en konvertering behövts.

I alla instanser av den här kombinationen av moduler, kombinationen av ”NXT Token” och ”NXT Query”, har frågan varit till för att hämta den data som scenariot ska hantera, exempelvis ”Products”.

#### 4.2.1.2 Stored Variables

En modul som används i samtliga scenarion är ”Stored Variables”. Denna modul är av slaget ”Data Store” som finns i Make för att få tillgång till de variabler som användaren, i detta fallet en utvecklare på Pector, har lagt in i sin Make-profil. I detta fall har det skapats en variabel vid namn ”CompanyNo” därför att GraphQL-frågorna kräver ett specifikt company number för att hämta data. Den variabeln har satts utanför alla scenarion för att endast behöva ändra den variabeln vid eventuell manipulation av programmet.

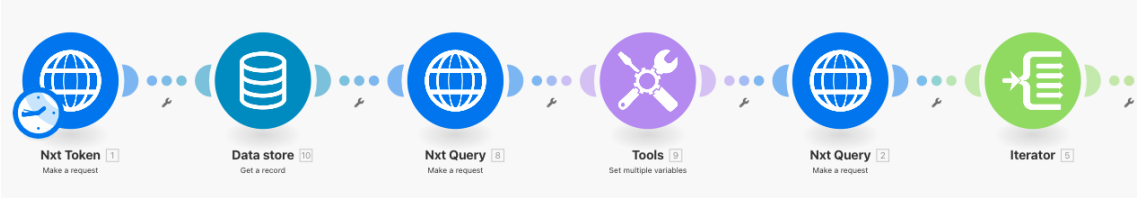
#### 4.2.1.3 Webhook

I orderscenariot har ett beslut tagits att använda en annan typ av modul som start i scenariot. De andra scenarierna startas nu manuellt vid behov, detta för att företaget inte har tagit ställning till hur scenarierna kommer att användas i framtiden. Däremot så behövde orderscenariot, enligt handledare, ha en mer omedelbar start när en händelse inträffar. Ordrar måste hanteras så fort de kommer in så att lagerstatus håller ett relevant värde och inte låter kunder beställa varor som inte finns på lager. Till detta har då Makes Webhook modul används.

Webhook-modulen ger utvecklaren möjligheten att starta scenarion när en händelse inträffar, som att en order har skapats i Shopify. Makes modul, samt Shopifys inställningar, gjorde det enkelt att sätta upp en koppling som signalerade till Make när en order har skapats. Detta gör att ”Order”-scenariot nu kan köras vid behov, när en order skapas.

### 4.2.2 Scenarier

I integrationen i Make används fem olika scenarion. De används för att föra över produkter, priser, kunder och lager från VBNXT till Shopify samt ordrar från Shopify till VBNXT. Samtliga scenarion, förutom den för ordrar, har samma typ av moduler i början. De sex första modulerna är av typen: ”Nxt Token - Make a request”, ”Data store - Get a record”, ”Nxt Query - Make a request”, ”Tools - Set multiple variables”, ”Nxt Query - Make a request” och ”Iterator”, se Figur 4.2.




**Figur 4.2:** De sex första modulerna som finns i scenarierna för produkter, priser, kunder och lager.

Den första modulen används för att få access till VBNXT och är identisk för de fyra scenarierna. Den andra modulen används för att hämta de variablerna som beskrivs i under-underavsnitt 4.2.1.2. Tredje och femte modulen används för att skicka en GraphQL-fråga till VBNXT, vilka är olika för varje scenario. Fjärde modulen skapar variabler från information som har hämtats och strukturerats om från VBNXT. Sjätte modulen är en iterator och används för att iterera över exempelvis produkter.

#### 4.2.2.1 Produkter

Scenariot för produkter framgår i Figur 4.3 och består av elva moduler. Modulerna är:

- ”Nxt Token - Make a request”
- ”Data store - Get a record”
- ”Nxt Query - Make a request”
- ”Tools - Set multiple variables”
- ”Nxt Query - Make a request”
- ”Iterator”
- ”Shopify - Search for Products”
- ”Shopify - Create a Product”
- ”Shopify - Search for Products”
- ”HTTP - Make a request”

- ”HTTP - Make a request”

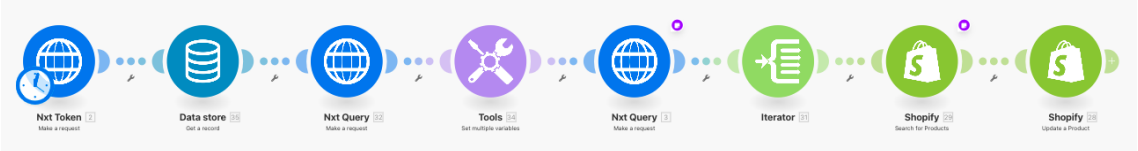
**Figur 4.3:** Make-scenario för att uppdatera information om produkter i Shopify.

Den andra modulen i Produkter används för att hämta variabeln för företagets id. Tredje modulen hämtar variabler från VBNXT som sedan grupperas i fjärde modulen. Femte modulen hämtar information om produkter i VBNXT. Första Shopify-modulen söker efter produkter som hittades i femte modulen. Produkter som finns i VBNXT men inte i Shopify går igenom filtret mellan första och andra Shopify-modulen medan produkter som redan finns i Shopify fastnar i filtret. Produkten skapas i Shopify och produkten får ett lager-id.

#### 4.2.2.2 Priser

Scenariot för priser framgår i Figur 4.4 och består av åtta moduler. Modulerna är:

- ”Nxt Token - Make a request”
- ”Data store - Get a record”
- ”Nxt Query - Make a request”
- ”Tools - Set multiple variables”
- ”Nxt Query - Make a request”
- ”Iterator”
- ”Shopify - Search for Products”
- ”Shopify - Update a Product”



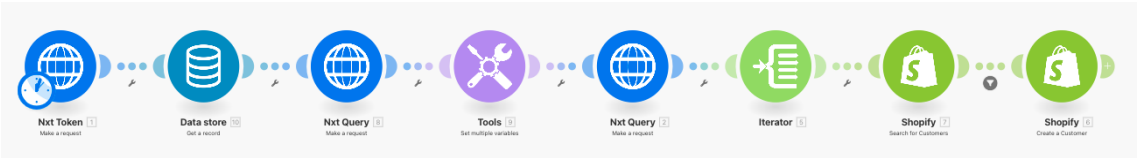

**Figur 4.4:** Make-scenario för att uppdatera produkters priser i Shopify.

Variabeln med företagets id hämtas i ”Data store”-modulen. De priser som är i rätt valuta samt de som är aktuella hämtas från VBNXT. Produkterna som finns i Shopify hämtas och varje produkt får sitt rätta pris.

#### 4.2.2.3 Kunder

Scenariot för kunder framgår i Figur 4.5 och består av åtta moduler. Modulerna är:

- ”Nxt Token - Make a request”
- ”Data store - Get a record”
- ”Nxt Query - Make a request”
- ”Tools - Set multiple variables”
- ”Nxt Query - Make a request”
- ”Iterator”
- ”Shopify - Search for Customers”
- ”Shopify - Create a Customer”




**Figur 4.5:** Make-scenario för att uppdatera information om kunder i Shopify.

Scenariot för kunder hämtar variabeln för företagets id. En GraphQL-fråga används för att hämta information om kunder i VBNXT. Kundens namn i VBNXT används för att söka i Shopify om kunden även finns där. Om kunden inte finns i Shopify så skapas en ny kund med det namnet i Shopify.

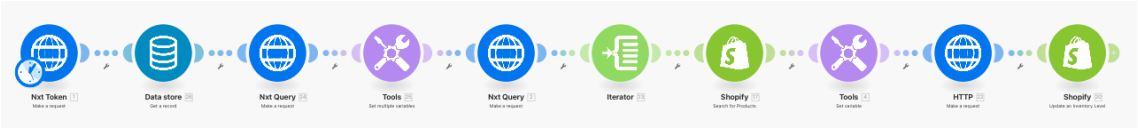
Det fanns inget fält för kund-id i Shopify medan det finns i VBNXT. Ett id för varje kund gör integrationen mer robust och minskar risken för fel under integrationen. Detta är något som hade behövts i Shopify.

#### 4.2.2.4 Lager

Scenariot för lager framgår i Figur 4.6 och består av tio moduler. Modulerna är:

- ”Nxt Token - Make a request”
- ”Data store - Get a record”
- ”Nxt Query - Make a request”
- ”Tools - Set multiple variables”
- ”Nxt Query - Make a request”
- ”Iterator”
- ”Shopify - Search for Products”

- ”Tools - Set variable”
- ”HTTP - Make a request”
- ”Shopify - Update an Inventory Level”




**Figur 4.6:** Make-scenario för att uppdatera lager på produkterna i Shopify

I scenariot för lager finns det en iterator som itererar över varje produkt. Antalet produkter i VBNXTs lager beräknas och läggs in i Shopify.

#### 4.2.2.5 Ordrar

Scenariot för ordrar framgår i Figur 4.7 och består av tolv moduler. Modulerna är:

- ”Webhooks - Custom webhook”
- ”Data store - Get a record”
- ”Nxt Token - Make a request”
- ”Nxt Query - Make a request”
- ”Router”
- ”Iterator”
- ”Nxt Query - Make a request”
- ”Shopify - Cancel an Order”
- ”Google sheets - Add a row”
- ”Email - Send an Email”
- ”Shopify - Get an Order”
- ”Nxt Query - Make a request”


**Figur 4.7:** Make-scenario för att skicka ordrar från Shopify till VBNXT

Scenariot för ordrar börjar med en Webhook. Detta innebär att scenariot exekveras varje gång en order skapas i Shopify. Den fjärde modulen innehåller en mutation för att skapa en order. Orderlinjer för ordern skapas i modulen ”Nxt Query” efter routern. Efter den modulen finns ett filter för att fånga felmeddelanden. Om ett felmeddelande upptäcks så upphävs ordern och felmeddelandet sparas. Priset läggs därefter till.

Routern som används i scenariot kör först den översta grenen av scenariot och sedan den andra grenen, när den första är klar.

# 5

# Resultat

De ursprungliga funktionella och icke-funktionella kraven som sattes upp i avsnitt 3.4 uppnåddes. Kravet om att hantera ordrar som lades till under projektets gång uppnåddes också. De ursprungliga kraven var realistiska så de uppnåddes innan planerad deadline. Det var därmed möjligt att i efterhand lägga till kravet om att hantera ordrar.

När de ursprungliga kraven för projektet var avklarade så hölls ett möte med de två handledarna för att redovisa innehållet. Då drogs slutsatsen att det är fördelaktigt för integrationen att lägga till funktionalitet för att hantera ordrar. Handledarna godkände kostnaden för integrationen eftersom vi redovisade hur mycket en operation för de olika scenarierna kostade i integrationen.

Det finns förbättringsområden för att få integrationen att fungera mer effektivt. Ett möjligt förbättringsområde är att synkronisera lagerstatus mellan Shopify och VBNXT så att lagerstatus bara hanteras på ett ställe, för att undvika att lagerstatus kan bli negativ. Kunden vill dels sälja produkter via Shopify och dels lägga in ordrar från kund direkt i VBNXT.

För att använda sig av Make i en integration krävs att man betalar en prenumeration. De finns i flera olika nivåer som tillåter olika antal operationer per månad. Om integrationen utvecklas så att många fler operationer krävs per scenario kan det leda till att kostnaden blir för hög. Då kan det vara värt att utforska andra lösningar för integrationen. Det kan då vara fördelaktigt att använda sig av .NET som är betydligt billigare än Make för stora integrationer.

# 6

# Slutsats

Syftet med arbetet var att implementera delar av en integration som överför data mellan Visma Business NXT och Shopify. Arbetet utfördes för företaget Pector AB som i början hade en kund för projektet. Kunden valde dock vid ett senare tillfälle att istället önska en integration till WooCommerce. Beslut togs att projektet fortfarande skulle fokusera på Shopify, då framtida kunder möjligen skulle kunna ha behov av en sådan lösning.

Make visade sig också vara en bra lösning för detta arbete. Tack vare dess användarvänlighet kunde arbetet drivas framåt i gedigen fart och problem kunde lösas snabbt. Problemet med hög kostnad visade sig inte vara relevant i detta fall, då integrationen enbart stod för en liten del av det stora arbetet, och en diskussion kring kostnad för företaget skulle föras vid ett senare skede. Däremot visade det sig att en lösning med .NET också troligtvis hade fungerat, med vissa begränsningar. Om en sådan lösning istället skulle ha implementerats så hade inte problemet med en för hög kostnad uppstått. Dock, eftersom Make innebar mindre arbete, valdes ändå detta som huvudprogrammet för arbetet.

Trots att .NET medför lägre kostnader för projekt som detta, förser Make utvecklare med unika verktyg som .NET i nuläget inte har. Makes användarvänliga gränssnitt och dess intuitiva design gör det mycket mer lättanvänt för utvecklare med mindre kunskap av att bygga applikationer. Eftersom detta är något som upplevdes tydligt under projektets gång, skulle det gå att dra slutsatsen att framtida utvecklingsverktyg har nytta av att gå i Makes riktning. Eftersom moduler redan till stor del är färdiga i gränssnittet reduceras kraven på programmeringsfärdighet, och utvecklare med mindre djup kunskap kan ta del av lika starka verktyg som erfarna programmerare.

Resultatet täckte de krav som diskuterades i början av projektet. Dataöverföringen sker på ett enkelt sätt är lätt att förstå samt enkel att utveckla vidare. Trots att det ursprungliga kundbehovet förändrades kunde integrationen följa, samt täcka, de krav som initialt sattes upp. Implementationen är anpassningsbar, iterativ och modulär. Pector kan med goda förutsättningar både utveckla och anpassa produkten utifrån deras kunders önskemål. Den medföljande dokumentationen består av viktig information samt tips för att föra arbetet vidare.

Programmet testades inte mot en verklig Shopify-affär. Detta kan ses som en möjlig begränsning då hänsyn inte tagits till problem som uppstår vid faktisk drift. Stora

mängder data kan medföra problem som kan förutses, vilket inte togs i beaktande för detta projekt. Dokumentationen har däremot tydligt beskrivit problemet med hög kostnad för Make och hur stora mängder data hade kunnat medföra negativa konsekvenser för kostnaden.

# Litteraturförteckning


[1] Traventus, "Välj rätt affärs- och ekonomisystem: Skillnaden mellan Visma Business och Business NXT," Available at https://www.traventus.se/valj-ratt-affars-och-ekonomisystem-skillnaden-mellan-visma-business-och-business-nxt/ (accessed on: 2025-04-14).

[2] Pector, "Slipp dubbelarbete och låt Business NXT göra jobbet," Available at https://www.pector.se/business-nxt/ (accessed on: 2025-06-12).

[3] Shopify, "Allt om Shopify Sverige," Available at https://www.shopify.com/se/om (accessed on: 2025-06-12).

[4] Make, "What is Make?" Available at https://help.make.com/what-is-make (accessed on: 2025-06-12).

[5] L. Byron, "GraphQL: A data query language," Available at https://engineering.fb.com/2015/09/14/core-infra/graphql-a-data-query-language/ (accessed on: 2025-05-20).

[6] Microsoft, "Introduction to .NET," Available at https://learn.microsoft.com/en-us/dotnet/core/introduction (accessed on: 2025-06-12).

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


CHALMERS