

# Erweiterungsentwicklung für Microsoft Dynamics 365 Business Central unter Verwendung von ExtensionsV2

Johannes Naderer  
(se0307)

# Inhaltsverzeichnis

<b>1</b>	<b>Kurzfassung</b>	<b>1</b>
<b>2</b>	<b>Abstract</b>	<b>2</b>
<b>3</b>	<b>Einleitung</b>	<b>3</b>
3.1	Motivation . . . . .	3
3.2	Zielsetzung . . . . .	4
<b>4</b>	<b>Stand der Technik</b>	<b>5</b>
4.1	ERP: Definition und Übersicht . . . . .	5
4.2	Geschichte und Grundarchitektur Dynamics 365 Business Central . . . .	7
4.2.1	Geschichte . . . . .	7
4.2.2	3-Schichten Architektur . . . . .	7
4.3	Objektarten in Business Central . . . . .	9
<b>5</b>	<b>Vergleich</b>	<b>12</b>
5.1	Aufgabenstellung . . . . .	12
5.2	Entwicklungsprozess . . . . .	13
5.2.1	Entwicklungsumgebung . . . . .	13
5.2.2	Grundfunktionalität . . . . .	16
5.2.3	Datenträgerexport . . . . .	17
5.2.4	Reporting . . . . .	17
5.2.5	Webservice-Anbindung . . . . .	17
<b>6</b>	<b>Tests und Evaluierung</b>	<b>18</b>
<b>7</b>	<b>Diskussion</b>	<b>19</b>
	<b>Quellenverzeichnis</b>	<b>20</b>
	Literatur . . . . .	20

Kapitel 1

Kurzfassung

## Kapitel 2

### Abstract

## Kapitel 3

# Einleitung

### 3.1 Motivation

ERP-Systeme sind heute aus dem wirtschaftlichen Umfeld nicht mehr wegzudenken [2] [1]. Bereits in den 1970er Jahren erkannten Wirtschaftstreibende Potential darin, ihre Prozesse und Unternehmensdaten zu digitalisieren. Während in den 1970er und 1980er Jahren innerhalb der einzelnen Abteilungen eines Unternehmens verschiedene Software-Lösungen zum Einsatz kamen, ist man sich mittlerweile einig, dass eine zentrale Applikation zur Verwaltung aller unternehmerisch wichtigen Daten und Prozessschritte große Vorteile liefert. Diese Erkenntnis hatte zur Folge, dass kleine Softwarehersteller immer mehr ins Wanken gerieten, da von der Wirtschaft allumfassende Software-Giganten gefordert wurden, die eine Vielzahl von Anforderungen aus den verschiedensten Anwendungsdomänen zu erfüllen haben. Solche Systeme können mit den meist begrenzten Ressourcen und Branchenwissen kleinerer Hersteller nicht entwickelt werden. Gleichzeitig entstanden durch die gewachsenen Anforderungen umfassende Softwaresysteme einiger größerer Hersteller, hier sind vor allem Marktführer SAP, aber auch Microsoft mit seiner Dynamics Sparte zu nennen.

Wer heute in einem Unternehmen mit der Einführung eines ERP-Systems betraut wird, muss sich intensiv mit den verschiedenen erhältlichen Lösungen auseinander setzen[2]. Denn neben Lizenzierung und finanziellen Aspekten, ist zu erarbeiten, welche Systeme die bestehenden Prozesse des Unternehmens am Besten abbilden. Systeme bilden Geschäftsprozess meist auf eine bestimmte Art ab. Sollte diese nicht mit dem Vorgehen des Unternehmens überein stimmen, bleiben meist nur zwei Auswegen offen. Entweder das Unternehmen passt seine Prozesse an die Vorgabe des Systems an, oder das System muss entsprechend angepasst werden, um den Ansprüchen des Unternehmensprozesses zu genügen.

Und genau hier spielt die Anpassbarkeit und Erweiterbarkeit eines Systems die zentrale Rolle. Gerade branchenspezifische und insbesondere unternehmensspezifische Prozesse müssen meist erst programmiert und in das System integriert werden. Programmierarbeiten und Änderungen am Standardsystem sind meist Aufwendig, und stellen so ein nicht zu vernachlässigendes finanzielles Risiko dar.

Um die Aspekte der Erweiterbarkeit und Anpassungsmöglichkeit möglichst gut zu erfüllen, entschied sich Microsoft im ERP-System Microsoft Dynamics NAV bereits in

sehr frühen Versionen, zertifizierten Entwicklern freien Zugang zum Applikationscode zu gewähren[3]. So können Entwickler die gesamte Geschäftslogik des Systems je nach Unternehmensanforderungen abändern und erweitern, in dem Sie neuen Programmcode hinzufügen, oder den Standardcode von Microsoft anpassen oder löschen. Dies hat zum Einen zur Folge, dass Entwickler mächtige Applikationen erstellen können, und sich diese direkt in das bestehende System integrieren lassen. Allerdings kommen mit diesen umfassenden Möglichkeiten auch Probleme auf. Je weiter die oft über Jahrzehnte verwendeten und erweiterten Systeme von der Codebasis von Microsoft abweichen, desto aufwendiger, fehleranfälliger und teurer ist es, diese Systeme mit den Aktualisierungen des Herstellers zu versorgen, die periodisch in das System integriert werden müssen.

Um die Systeme update-fähig zu halten, und gleichzeitig ein Entwicklungsmodell zu schaffen, dass auch in der Cloud-Variante des ERP-Systems funktionieren kann, wurde mit Dynamics NAV 2017 erstmals das Konzept der Erweiterungsprogrammierung mit ExtensionsV1 für Dynamics NAV vorgestellt. ExtensionsV1 ist ein gänzlich neuer Ansatz für das ERP-System zu programmieren, und hat konzeptionell viele Vorteile gegenüber der konventionellen prozeduralen Entwicklung, ist aber mittlerweile aufgrund einiger technischer Schwierigkeiten obsolet.

Mit der in 2018 veröffentlichten Version - ExtensionsV2 - sind nicht nur viele der technischen Mängel behoben, ExtensionsV2 kommt auch mit einer neuen Programmiersprache und Entwicklungsumgebung.

## 3.2 Zielsetzung

Im Rahmen dieser Arbeit, wird ein Überblick über die Plattform Dynamics NAV bzw. die Cloud-Variante Dynamics 365 Business Central und die Programmierung dieser Systeme gegeben. Hierfür wird erst eine Übersicht über das Gesamtsystem und seine Schichtenarchitektur vermittelt. Anschließend wird das Konzept der Erweiterungsentwicklung mit ExtensionsV2 mit der konventionellen prozeduralen Entwicklung verglichen. Dies erfolgt anhand eines Beispiels, dass auf beide Arten gelöst wird. Einerseits wird die Anwendung in der Entwicklungsumgebung C/SIDE mit C/AL entwickelt. Andererseits wird anhand der Aufgabenstellung eine Erweiterung mit ExtensionV2 in VisualStudio Code und der aktuellen Sprache AL erstellt. Hierbei liegt der Fokus nicht darauf, kleine syntaktischen Unterschiede zwischen den Sprachen hervorzuheben, sondern Neuerungen in der Sprache AL zu beleuchten, und konzeptionelle Unterschiede zwischen den beiden Programmierparadigmen aufzuzeigen und zu bewerten.

Der Vergleich erfolgt anhand von statischen und dynamischen Code-Metriken sowie Laufzeitmessungen. Zusätzlich wird auch diskutiert, welche Vor- und Nachteile sich durch die nun neue Datei-basierte Codeverwaltung hinsichtlich der Einbindung und Nutzung von Source Code Management und Continuous Integration Systemen ergeben. In einem letzten Block wird danach das Event-basierte Programmiermodell der Erweiterungsentwicklung mit ExtensionsV2 diskutiert, die Vor- und Nachteile beleuchtet, die mit dem Wechsel von Code-Anpassung hin zu Code-Erweiterung einher gehen.

## Kapitel 4

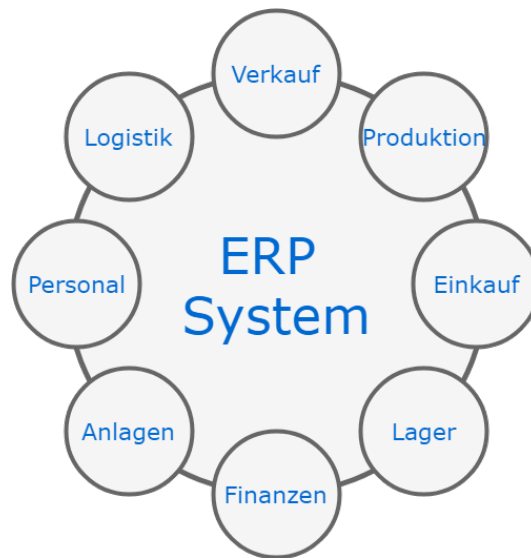
# Stand der Technik

### 4.1 ERP: Definition und Übersicht

ERP Systeme sind umfangreiche kommerzielle Softwaresysteme, mit der Kernaufgabe, alle Abteilungen und Prozesse eines Unternehmens soweit wie möglich digital abzubilden[6]. Ein ERP-System stellt für ein Unternehmen somit eine zentrale Verarbeitungs- und Datensicherungsplattform für sämtliche unternehmensrelevanten Geschäftsdaten bereit. Die Daten sind hierbei in einem einzelnen System erfasst, und sind so sofort für alle Unternehmensbereiche verfügbar. Es sind keine Synchronisierungsschritte oder Schnittstellen innerhalb des Systems nötig. Durch die zentrale Datenerfassung, stellt ein ERP-System eine durchgängige Informationsquelle für alle Unternehmensbereiche dar, die bei der Prozessanalyse und Analyse als Datenbasis für geschäftliche Entscheidungsträger unabdingbar ist.

Im Gegensatz zu abteilungsbezogenen Systemstrukturen (Insellösungen) ist es durch Einsatz eines ERP-Systems möglich Funktionen zu nutzen, für deren Durchführung Informationen aus mehreren Abteilungen nötig sind[5]. So kann zum Beispiel bei Eingang eines Auftrags sofort automatisiert geprüft werden, ob der Auftrag angenommen werden soll. Entscheidungen wie diese basieren auf einem sehr breiten Datenstamm aus den verschiedenen Abteilungen. Aus den Daten der Finanzabteilung können Zahlungsmoral, offene Beträge des Kunden und ein voraussichtlicher Deckungsbeitrag eine Rolle für diese Entscheidung spielen. Anhand der Lagerhaltungsdaten kann sofort eine Verfügbarkeitsprüfung für die bestellten Artikel durchgeführt werden. Mithilfe von Produktions- und Personaldaten wird ausgewertet, ob ausreichend Personal und Maschinenressourcen für die Erfüllung des Auftrags zur Verfügung stehen. Dies sind nur einige wenige Beispiele, wie ein ERP-System bei der täglichen unternehmerischen Tätigkeit behilflich sein kann.

Da in ERP-Systemen der Zugriff auf die Datenbank nicht durch die Systemarchitektur eingeschränkt ist, muss der Zugang zu den Daten im System über ein Rechte- und Modulsystem gesteuert werden[5]. Berechtigungssätze lassen sich hier meist sehr feingranular definieren, sodass einerseits der Schutz sensibler Daten gewährleistet ist, jedoch andererseits alle benötigten Daten entsprechend betrachtet und verarbeitet werden können.



**Abbildung 4.1:** Schematische Darstellung: Modularisierung anhand Unternehmensabteilung

Um die umfangreichen Funktionalitäten eines ERP-Systems zu gliedern und aufzuteilen, bedienen sich die meisten Hersteller eines Modul-Systems. Meist spiegelt die Aufteilung dieser Module die einzelnen Abteilungen eines Unternehmens wieder. So verteilt sich die Gesamtfunktionalität eines Systems beispielsweise auf ein Einkaufsmodul, ein Vertriebsmodul und viele andere Teilbereichsmodule auf. Diese Module können in ihren Grundzügen unabhängig voneinander verwendet werden. So kann ein Unternehmen beispielsweise entscheiden, vorerst nur Finanzen und Personal über das System zu verwalten. Andere Module können im Laufe der Zeit stückweise in Betrieb genommen werden. Zu den bekanntesten ERP Herstellern zählen unter anderen IBM, SAP, Microsoft, Infor und Sage.



## 4.2 Geschichte und Grundarchitektur Dynamics 365 Business Central

### 4.2.1 Geschichte

Das ERP-System, das heute *Microsoft Dynamics 365 Business Central* heißt, erschien ursprünglich 1984 unter dem Namen *PCPlus* als ein ERP System für Microsoft DOS in Dänemark[4]. Während seiner mittlerweile 35-jährigen Geschichte wurde das Produkt einige Male neu benannt und an den technischen Fortschritt angepasst. Was 1984 begann, wurde 1995 unter dem Namen *Navision Financials* als das erste ERP-Produkt mit grafischer Benutzeroberfläche für Windows95 präsentiert. 2002 wurde *Navision Financials* von Microsoft gekauft und unter dem Namen *Microsoft Business Solutions Navision* vertrieben. In all diesen Jahren basierte die Datenspeicherung des Systems in einem komplexen Dateibasierten Format. Im Jahr 2008 passiert dann der Schritt zu Microsoft SQL Server und der 3-Schichten-Architektur, nun unter dem Namen *Microsoft Dynamics NAV 2009*. Der vorerst letzte Meilenstein in der Geschichte des Systems ist 2018. Das System wird nun als Cloud-ERP-System unter dem Namen *Microsoft Dynamics 365 Business Central* betrieben.

Trotz den vielen Versionen und der jahrzehntelangen Geschichte dieses Systems, finden sich auch in der heutigen Code-Basis noch viele Passagen, die bereits in den 1980er Jahren entstanden, und bis heute produktiv eingesetzt werden.

### 4.2.2 3-Schichten Architektur

Dynamics 365 Business Central basiert auf einer 3-Schichten Architektur. Durch Schichtenarchitekturen lassen sich die Aufgabengebiete bzw. Teile eines komplexen Softwaresystems aufteilen. Im Falle von Dynamics Business Central 365 unterscheiden wir zwischen der Endbenutzerschicht, Serverschicht und Datenbankschicht.

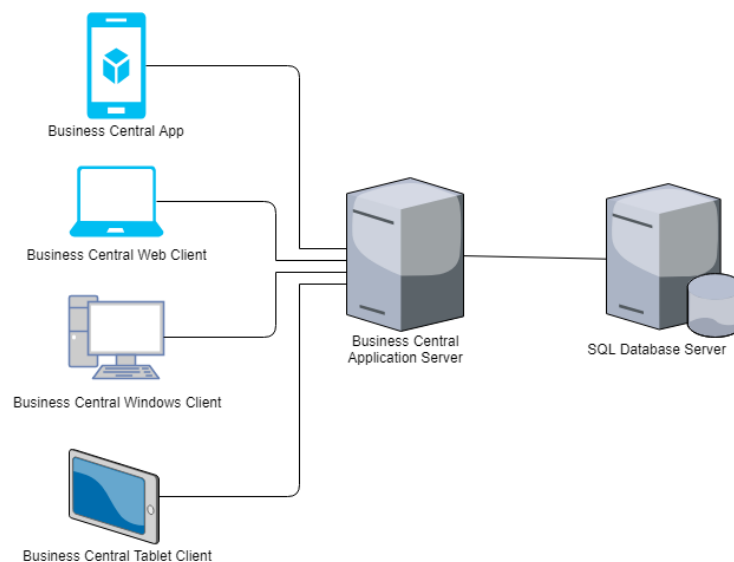


Abbildung 4.2: Dynamics 365 Business Central: 3-Schichten Architektur

Schicht 1: Die Endbenutzerschicht/Präsentationsschicht: In dieser Schicht der Architektur finden sich sämtliche Softwarekomponenten, die direkt die von der Serverschicht exportierten Funktionalitäten nutzen. Hierzu zählen vorrangig die von Microsoft veröffentlichten Endbenutzerprogramme wie der Business Central WebClient und die Business Central Mobile App. Aber auch von Drittanbietern erstellte Softwarekomponenten, die die Microsoft Graph API oder von Business Central veröffentlichte Webdienste nutzen sind Teil der Endbenutzerschicht.

Schicht 2: Die Serverschicht: Die Business Central Serverapplikation (auch *Middle-Tier* oder *Service-Tier*) stellt das Herzstück des Gesamtsystems dar. Der Business Central Server ist eine .NET basierte Serveranwendung und nutzt die Windows Communication Foundation (WCF) als Kommunikationsprotokoll. Der Server nimmt sämtliche Anfragen von Endbenutzerprogrammen entgegen, holt anhand dieser Anfragen Daten von der Datenbankschicht ab, führt mithilfe der abgeholten Daten Geschäftslogik aus, bereitet die Ergebnisse der Geschäftslogik auf, und liefert diese zurück an das anfragende Endbenutzerprogramm. Neben den Endpunkten zur Client-Kommunikation beinhaltet die Serverkomponente auch die Webserverkomponenten zur Nutzung des WebClients. Die Webserverkomponente selbst ist eine ASP.NET Core Applikation, die auf einem mitgeliefertem IIS (Internet Information Server) läuft. Daher ergibt sich auch die Anforderung, dass die Serverschicht auf einer Windows-Server Maschine betrieben werden muss.

Schicht 3: Die Datenbankschicht: Hinter der Datenbankschicht verbirgt sich eine Microsoft SQL Server Instanz. Hierbei ist zu erwähnen, dass aufgrund der historischen Entwicklung des Gesamtsystems einige Funktionen einer klassischen relationalen Datenbank hier nicht verwendet werden. So wird man am Datenbankserver vergeblich nach Relationen zwischen Tabellen suchen (Fremdschlüsselbeziehung), denn diese existieren hier schlichtweg nicht. Diese Beziehungen werden von der Serverschicht verwaltet. Auf Grund dieser Tatsache ist es strengstens abzuraten manuell mit SQL Befehlen Datenbestände zu ändern. Änderungen, die nicht durch die Logik der Serverschicht validiert werden können, können schnell zu Inkonsistenzen in den Daten führen, die in weiterer Folge das Gesamtsystem korrumpieren und zu Systemausfällen führen können.

Die Unterteilung der einzelnen Teilbereiche des Systems in drei Teilbereiche liefert einige Vorteile. So können anhand der vom Server exportierten Schnittstellen schnell neue Apps und

### 4.3 Objektarten in Business Central

Die Programmierung von Microsoft Dynamics 365 Business Central erfolgt durch die Erstellung und Anpassung von Applikationsbauteilen die gemeinhin *Objekte* genannt werden. Diese *Objekte* sind nicht mit jenen aus der klassischen objektorientierten Programmierung zu vergleichen. Dynamics 365 ist objektbasiert und nicht objektorientiert. Entwickler haben nicht die Möglichkeit neue Typen von Objekten zu erstellen, sondern nur neue Ausprägungen der bestehenden Objekttypen zu entwickeln. AL bietet gegenüber C/AL neue Möglichkeiten zur Entwicklung unter AL, was sich auch in den verfügbaren Objekttypen ausdrückt.

Type	C/AL	AL
Table	X	X
TableExtension		X
Page	X	X
PageExtension		X
Report	X	X
Codeunit	X	X
Query	X	X
XMLPort	X	X
MenuSuite	X	
Enum		X

**Tabelle 4.1:** Verfügbarkeit Objekttypen: C/AL und AL

**Table:** Tabellenobjekte definieren den Datenaufbau, Restriktionen und Validierungsregeln für alle Daten in Microsoft Dynamics 365 Business Central. Beim Kompilieren von Tabellenobjekten, veranlasst die Serverschicht die Erstellung oder Änderung einer zum Tabellenobjekt gehörigen SQL-Tabelle, in der schlussendlich die Daten landen. Tabellenobjekte sind aber weit mehr als lediglich eine Beschreibung zur Datenhaltung. Tabellenobjekte definieren auch sämtliche Validierungslogik, sowohl auf Feld- als auch auf Datensatzebene. Darüber hinaus gibt es auch die Möglichkeit, innerhalb von Tabellenobjekten Funktionen und Prozeduren für die beschriebenen Daten zu definieren.

**TableExtension:** Tabellenerweiterungen sind ein zentraler Baustein der Erweiterungsentwicklung mit AL. Sollten an einer Tabelle Änderungen oder Erweiterungen nötig sein, würde man mithilfe C/AL einfach das bestehende Tabellenobjekt abändern. Mit dem Konzept der Erweiterungsentwicklung und AL ist dies nicht mehr möglich. Genau hier kommen Tabellenerweiterungsobjekte ins Spiel. Mithilfe von Tabellenerweiterungen lassen sich Felder und Logiken eines Tabellenobjektes erweitern, ohne das Tabellenobjekt selbst zu ändern. Beim Kompilieren von Tabellenerweiterungen wird auf SQL-Seite zusätzlich zur Basistabelle eine zusätzliche hinzugefügt (textitCompanion Table). Diese *Companion Table* verfügt über den selben Primärschlüssel wie die Basistabelle, und bietet Platz für Felder der Erweiterung. Auf SQL-Seite wird für die Benennung der *Companion Table* der Name der Basistabelle um die eindeutige GUID der Erweiterung

ergänzt. Tabellenerweiterungen sind nur unter AL verfügbar.

dbo.CRONUS AT\$Cust_ Ledger Entry				dbo.CRONUS AT\$Cust_ Ledger Entry\$3d5b2137-efeb-4014-8489-41d37f8fd4c3			
PK	Entry No_	Customer No_	Posting Date	PK	Entry No_	ExtensionField1	ExtensionField2

**Abbildung 4.3:** Dynamics 365 Business Central: Standardtabelle und dazugehörige Tabellenerweiterung

**Pages:** Seiten sind unter Dynamics 365 Business Central der Baustein zur Erstellung grafischer Benutzeroberflächen. Im Vergleich zu anderen Systemen hat man hier jedoch keinen bedeutenden kreativen Freiraum bei der Gestaltung. Seiten definieren lediglich die angezeigte Information, deren Gruppierung, Sortierung und die anwendbaren Aktionen auf diese. Die visuelle Gestaltung wird bis auf wenige Ausnahmen vom System vorgegeben, sodass man als Entwickler in diesem Aspekt nur sehr eingeschränkte Möglichkeiten hat. Diese Art der Einschränkung ist zwar auf den ersten Blick als Nachteil zu betrachten, garantiert dem Nutzer des Systems jedoch ein durchgängiges Design der Benutzeroberflächen.

**PageExtension:** Mithilfe von Seitenerweiterungen lassen sich bestehende Seiten anpassen und ergänzen, ohne das ursprüngliche Seitenobjekt abzuändern. Der Benutzer merkt bei Betrachtung des Ergebnisses nicht, dass es sich hierbei um ein zusätzliches Objekt handelt. Rein optisch integrieren sich Seitenerweiterungen nahtlos in ihre Ursprungsobjekte.

**Report:** Berichte erlauben es, Auswertungen und Dokumente aus dem System zu generieren, etwa eine Bilanzübersicht, oder eine Verkaufsrechnung. Dabei bestehen Berichte aus zwei Komponenten: einem Dataset und einem Layout. Im Dataset werden die im Layout zur Verfügung stehenden Daten definiert. Das Layout selbst bestimmt die grafische Repräsentation dieser. Durch die Integration von Datasets in Microsoft Word Vorlagen, lassen sich durch den Nutzer auf einfache Weise benutzerdefinierte Word-Layouts erstellen.

**Codeunit:** Codeunits stellen die Logikkomponenten des Systems dar. Sämtliche Berechnungen, Buchungsroutinen und andere Teile der Geschäftslogik finden sich hier. Um die Funktionalität einer Geschäftslogik abzuändern, ist es unter C/AL üblich direkt den verantwortlichen Code dafür abzuändern. Unter AL können bestehende Codeunits nicht geändert werden. Um unter AL Änderungen durchzuführen, müssen Event-Subscriber erstellt werden, die sich in die Standardlogik einklinken.

**Query:** Abfrageobjekte bieten die Möglichkeit, hochperformante Abfragen an die Datenbank abzusetzen. Die Ergebnisse dieser Abfragen werden meist als Dataset für Berichte genutzt, oder als Webservice exportiert.

XMLPort: XMLPorts bieten eine einfache Möglichkeit, Daten in das System zu importieren und aus dem System zu exportieren. Trotz des Objektnamens, sind XMLPorts nicht nur auf XML beschränkt, sondern können auch verwendet werden um CSV und andere Formate zu verarbeiten.

MenuSuite: MenuSuite-Objekte bestimmen die hierarchische Menüführung des Systems, und bestimmen wie andere Objekte über die grafische Oberfläche aufgerufen werden können. MenuSuites sind nur unter C/AL verfügbar, unter AL werden die nötigen Informationen dafür direkt in den Seiten- und Reportobjekten selbst definiert.

Enum: Nur unter AL verfügbar. Enums definieren Aufzählungstypen und ersetzen den in C/AL verwendeten *Option* Datentyp, der lediglich eine kommasperierte Zeichenfolge darstellt, schlecht erweiterbar ist, und somit für die Erweiterungsentwicklung unter AL nicht zielführend ist.

## Kapitel 5

# Vergleich

Folgend wird zur Darstellung des Entwicklungsprozesses, und der Unterschiede zwischen der konventionellen prozeduralen Entwicklung in C/AL, und der erweiterungsbasierten Entwicklung in AL ein Beispiel in beiden Sprachen entwickelt.

### 5.1 Aufgabenstellung

Für die Kunden des Auftraggebers unserer Erweiterung sollen Treuepunkte verwaltet werden. Treuepunkte werden mit dem Kauf von Waren verdient, oder von der Marketingabteilung an Bestandskunden vergeben. Treuepunkte können beim Kauf von Produkten eingelöst werden, um einen Preisnachlass zu erzielen. Eingelöste Punkte verringern den Rechnungsbetrag um einen bestimmten Geldwert, der variieren kann. So mag ein Treuepunkt im Januar 10 Cent wert sein, im Februar jedoch 15 Cent. Die Schwankung des Treuepunktwertes wird als Marketinginstrument genutzt. Auch wie viele Treuepunkte beim Einkauf vergeben werden ist variabel, so sind etwa Aktionszeiträume vorgesehen, in denen beim Einkauf doppelt so viele Treuepunkte verdient werden können.

Das neue Treuepunktesystem ist für das Marketing von hoher Bedeutung. So ist es erforderlich, dass Änderungen am Treuepunktekonto eines Kunden über einen Web Service an das verwendete CRM-System gemeldet werden. Für das Reporting im Unternehmen ist es außerdem nötig, dass täglich ein XML Datenträger erzeugt werden kann, in dem der Treuepunktesaldo und die Bewegungen des aktuellen Tages je Kunde ersichtlich sind. Zusätzlich zu dieser Datei für das Berichtssystem soll auch ein übersichtlicher Ausdruck in PDF-Form an die Marketingleitung gesendet werden.

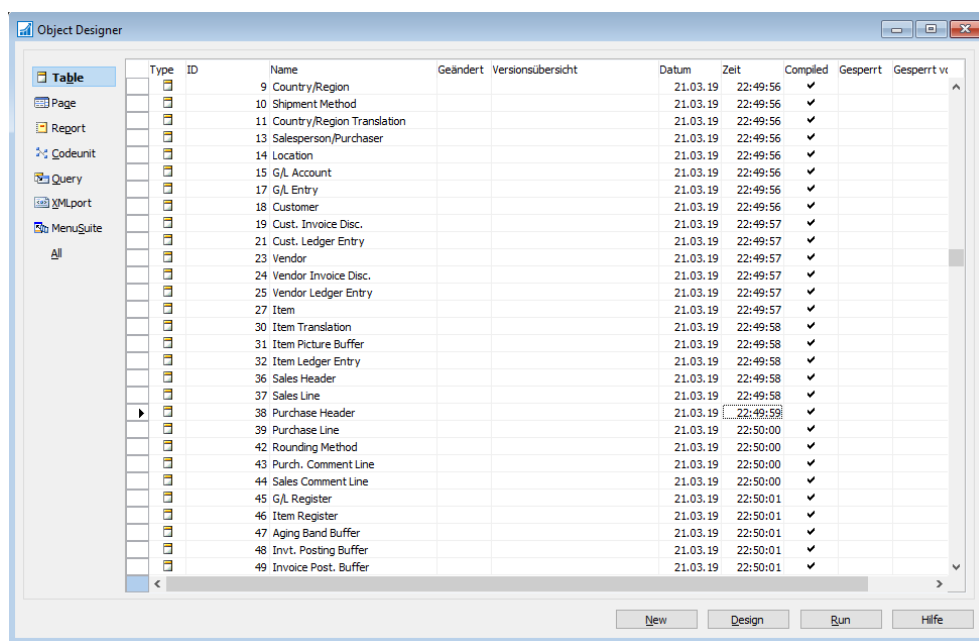
## 5.2 Entwicklungsprozess

### 5.2.1 Entwicklungsumgebung

#### C/AL - Development Environment

C/AL wird im *Microsoft Dynamics Development Environment* entwickelt. Dabei handelt es sich eigentlich um den Client, der bis zur Version 2009 noch als Windows Client für Endbenutzer verwendet wurde, nun seit dem jedoch rein für die Entwicklung genutzt wird. Das Development Environment ist ein Windows Client, der stets sowohl mit Datenbank, als auch mit der Serverapplikation verbunden sein muss. Die Datenbankverbindung ist nötig, da darin die Applikationsobjekte gespeichert sind, die Verbindung zum Applikationsserver, um Änderungen kompilieren und ausführen zu können.

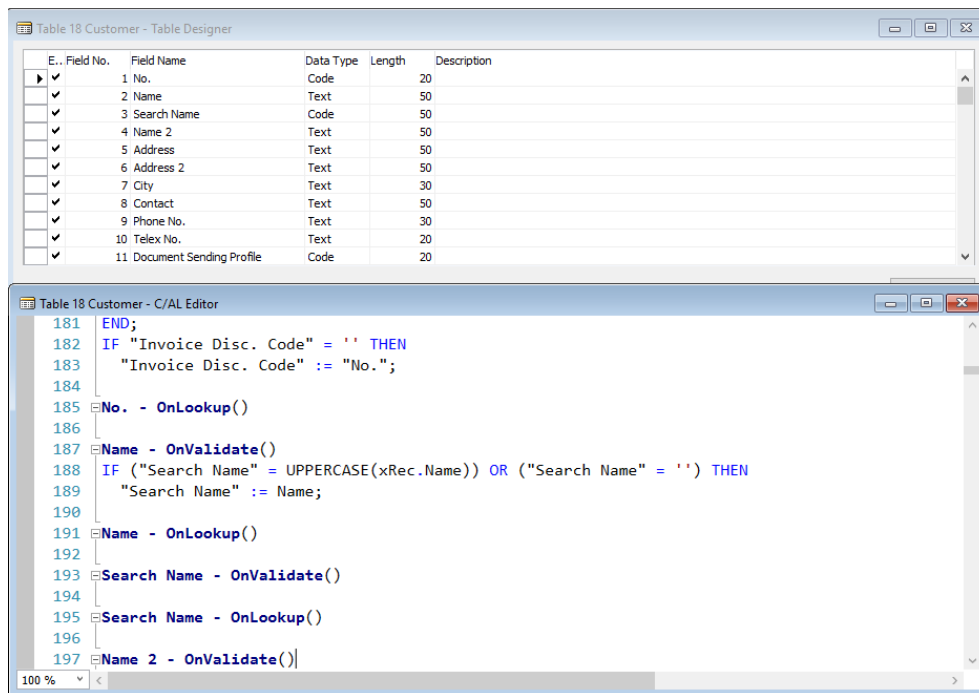
Das Kernstück des Development Environment bildet der *Object Designer*. Der *Object Designer* liefert einen Überblick über sämtliche, im System vorhandenen Applikationsobjekte und Details zu Ihnen. Ein Applikationsobjekt unter C/AL wird durch eine numerische ID und seinen Namen identifiziert. Zusätzlich wird zu den einzelnen Applikationsobjekten auch gespeichert, ob und wann sie das letzte Mal geändert wurden.



Type	ID	Name	Geändert	Versionsübersicht	Datum	Zeit	Compiled	Gesperrt	Gesperrt v.
Table	9	Country/Region			21.03.19	22:49:56	✓		
Table	10	Shipment Method			21.03.19	22:49:56	✓		
Table	11	Country/Region Translation			21.03.19	22:49:56	✓		
Table	13	Salesperson/Purchaser			21.03.19	22:49:56	✓		
Table	14	Location			21.03.19	22:49:56	✓		
Table	15	G/L Account			21.03.19	22:49:56	✓		
Table	17	G/L Entry			21.03.19	22:49:56	✓		
Table	18	Customer			21.03.19	22:49:56	✓		
Table	19	Cust. Invoice Disc.			21.03.19	22:49:57	✓		
Table	21	Cust. Ledger Entry			21.03.19	22:49:57	✓		
Table	23	Vendor			21.03.19	22:49:57	✓		
Table	24	Vendor Invoice Disc.			21.03.19	22:49:57	✓		
Table	25	Vendor Ledger Entry			21.03.19	22:49:57	✓		
Table	27	Item			21.03.19	22:49:57	✓		
Table	30	Item Translation			21.03.19	22:49:58	✓		
Table	31	Item Picture Buffer			21.03.19	22:49:58	✓		
Table	32	Item Ledger Entry			21.03.19	22:49:58	✓		
Table	36	Sales Header			21.03.19	22:49:58	✓		
Table	37	Sales Line			21.03.19	22:49:58	✓		
Table	38	Purchase Header			21.03.19	22:49:59	✓		
Table	39	Purchase Line			21.03.19	22:50:00	✓		
Table	42	Rounding Method			21.03.19	22:50:00	✓		
Table	43	Purch. Comment Line			21.03.19	22:50:00	✓		
Table	44	Sales Comment Line			21.03.19	22:50:00	✓		
Table	45	G/L Register			21.03.19	22:50:01	✓		
Table	46	Item Register			21.03.19	22:50:01	✓		
Table	47	Aging Band Buffer			21.03.19	22:50:01	✓		
Table	48	Invt. Posting Buffer			21.03.19	22:50:01	✓		
Table	49	Invoice Post. Buffer			21.03.19	22:50:01	✓		

Abbildung 5.1: Development Environment: Object Designer und C/AL Editor

Je nach ausgewählter Objektart stellt das Development Environment einen auf die Objektart angepassten *Designer* zur Verfügung, über den bereits einige Basiseinstellungen getätigt werden können. Im Falle von Tabellenobjekten, können mithilfe des Table Designers Tabellenfelder angelegt, entfernt und bearbeitet werden. Über den Designer gelangt man ebenfalls zum C/AL Editor, in dem die Implementierung der Geschäftslogik passiert.



**Abbildung 5.2:** Development Environment: Table Designer und C/AL Code Editor der Debitoren Tabelle

Die Sprache C/AL basiert auf Pascal. Im Gegensatz zu Pascal ist C/AL jedoch rein prozedural, und rein auf die Arbeit mit Dynamics NAV bzw. Business Central spezialisiert. So bietet C/AL mithilfe der inkludierten *Record API* eine einfache und effiziente Weise, Datensätze aus der Datenbank zu lesen, filtern, schreiben und zu löschen. Der Mehraufwand der in anderen Sprachen und Systemen durch die Erstellung von Datenbankverbindungen verursacht wird ist in C/AL minimal, da die Datenbankverbindung bereits durch die Verbindung zum Server gegeben ist. Der Datenbankkontext kann daher vom Applikationsserver bestimmt werden, und muss nicht im Applikationscode definiert werden. Andererseits fehlen innerhalb C/AL Funktionalitäten, die in modernen Programmiersprachen mittlerweile zum Standardumfang fehlen, wie zum Beispiel eine Möglichkeit zur Kommunikation via HTTP.

Um Applikationsobjekte zwischen verschiedenen Datenbanken zu transferieren, um beispielsweise entwickelte Applikationsobjekte von einem Testsystem in die Produktivumgebung zu übernehmen, bietet das Development Environment die Möglichkeit Applikationsobjekte zu exportieren. Dieser Export kann in zwei verschiedenen Formaten erfolgen. Einerseits im Textformat. Dabei werden sowohl Code als auch die im Designerfenster getätigten Einstellungen in ein spezielles Textformat gebracht. Dieses Textformat (Dateiendung .txt) ist zwar grundsätzlich durch den Menschen lesbar, definiert jedoch auch einige intern nötige Eigenschaften. Andererseits können Objekte auch im Binärformat exportiert werden (Dateiendung .fob). Das Binärformat zeichnet sich im Gegensatz zum Textformat durch geringere Dateigröße und besserer Performanz beim Importie-



ren und Exportieren aus, und ist daher das Standardformat um Applikationsobjekte zwischen Datenbanken zu transferieren.

### AL - Visual Studio Code

Die Programmierung von Erweiterungen für Microsoft Dynamics 365 Business Central erfolgt in Visual Studio Code [7]. Visual Studio Code ist ein OpenSource Quelltext-Editor für verschiedenste Programmier- und Markupsprachen basierend auf dem Electron Framework. Visual Studio Code ist in der Sprache Typescript implementiert. Im Gegensatz zum Development Environment setzt Visual Studio Code kein Windows-Betriebssystem voraus, sondern kann auch unter Mac und Linux verwendet werden. Als zeitgemäße Entwicklungsumgebung liefert Visual Studio Code eine Auswahl einiger Features für Entwickler, die im Development Environment nicht vorzufinden sind. Darunter:

- Refactoring Werkzeuge
- IntelliSense und Code-Vervollständigung
- Mauslose Bedienung
- Integrierte Source Code Verwaltung mit Git
- Erstellung von benutzerdefinierten Tastenkombinationen

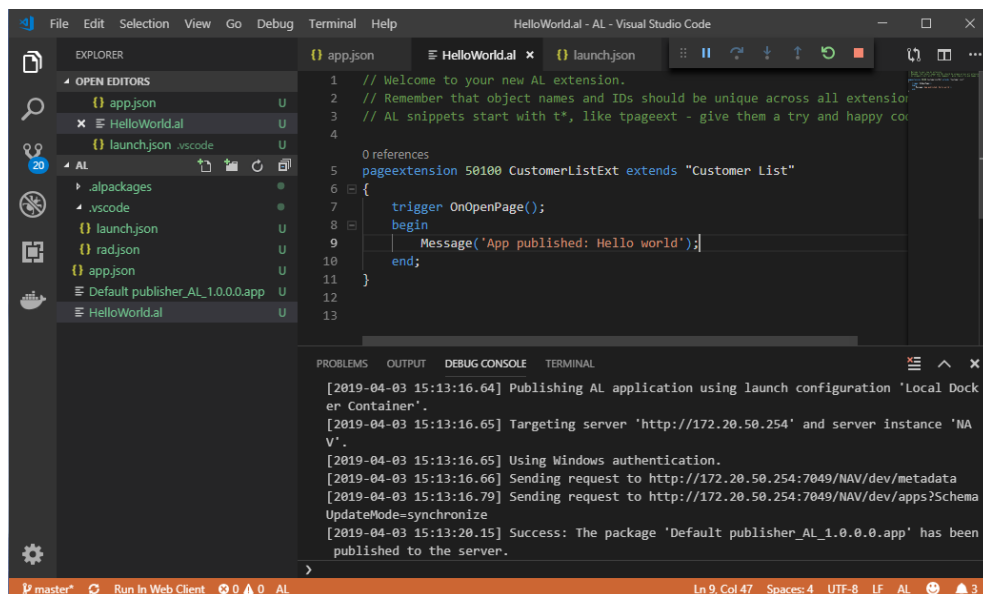


Abbildung 5.3: Visual Studio Code: Grafische Oberfläche, Hello World Extension

Im Gegensatz zum Development Environment ist Visual Studio Code nicht dafür ausgelegt, mit Business Central und AL zu Arbeiten. Eine Basisinstallation von Visual Studio Code kann so auch nicht für die Entwicklung unter AL genutzt werden. Hier kommt jedoch eine große Stärke von Visual Studio Code ins Spiel, seine Erweiterbarkeit. Mit Business Central wird die dazugehörige Visual Studio Code Erweiterung mitgeliefert, die für die Entwicklung von AL-Erweiterungen nötig ist. Diese Erweiterung *AL Language Extension*, wird im .vsix Format von Business Central zur Verfügung gestellt und lässt sich mittels weniger Klicks installieren.

Visual Studio Code wird monatlich automatisch mit Updates versorgt. Auch Neuheiten für die AL Spracherweiterung werden automatisch mitinstalliert, wobei es einfach möglich ist, frühere Versionen der Erweiterung zu verwenden um auch mit Systemen arbeiten zu können, die noch nicht auf dem neuesten Stand sind. Dies stellt für Entwickler einen bedeutenden Vorteil dar, da das Development Environment bei Neuerungen immer manuell geladen werden musste, und mehrere lokale Installationen nötig waren, um auch vorangegangene Versionen des Systems zu unterstützen.

Visual Studio Code in Kombination mit AL ist rein textbasiert. Die aus dem Development Environment bekannten verschiedenen Designer Fenster finden unter AL keine Anwendung mehr. Applikationsobjekte werden nicht mehr direkt aus der Datenbank gelesen und zurückgeschrieben, sondern existieren nun zur Entwicklungszeit als Dateien in einem Verzeichnis auf der Entwicklerrmaschine. Somit sind keine proprietären Exportmechanismen mehr nötig, die Datei beinhaltet sämtliche Informationen für das spätere Laufzeitobjekt, und wird als solches komplett vom Entwickler verfasst. Im Gegensatz zum Textexport aus dem Development Environment steht in den erstellten AL Dateien genau was der Entwickler vorgibt. Nicht mehr und nicht weniger. Dies ist einer der größten Vorteile, die die neue Entwicklungsumgebung mit sich bringt. Denn dadurch lässt sich der geschriebene Quellcode sinnvoll und ohne Umwege in einem Source Code Management System wie Git verwalten.

### 5.2.2 Grundfunktionalität

Um die Grundfunktionalität der Treuepunkterweiterung, das Sammeln und Einlösen von Treuepunkten abzubilden, sind einige Änderungen und Ergänzungen an der Standard-Tabellenstruktur von Dynamics 365 Business Central nötig.

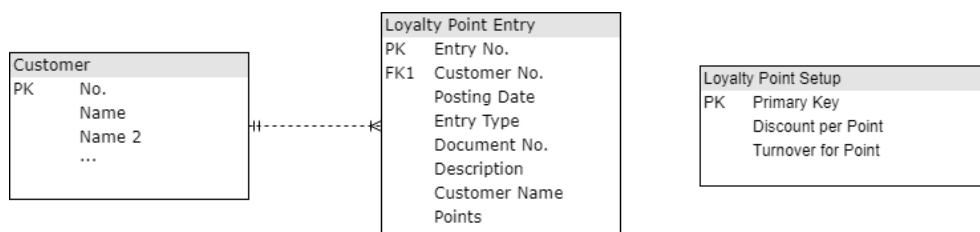


Abbildung 5.4: Grundfunktionalität: Tabellen

Es wird eine neue Tabelle *Loyalty Point Entry* eingeführt, in der alle Transaktionen

betreffend Treuepunkten gespeichert werden. Um diverse Auswertungen zu ermöglichen, werden neben der betroffenen Punktezahl, und dem zugehörigen Kunden auch die Dokumentennummern und die Transaktionsart (*Entry Type*) gespeichert. Die Transaktionsart kann dabei einen von drei Werten annehmen: Verdienst, Einlösung und Marketing. Um die Treuepunkterweiterung konfigurierbar zu machen wird zusätzlich auch eine Setup-Tabelle *Loyalty Point Entry* erstellt. Diese folgt dem in Dynamics 365 Business Central oft gebrauchten Softwaremuster der Setup-Tabelle. Dabei handelt es sich um eine Tabelle mit einem Primärschlüssel *Primary Key*, in der maximal ein Datensatz gespeichert werden kann. In Tabellen dieser Art werden Konfigurationen getroffen, um andere Bereiche der Geschäftslogik zu parametrisieren.

- *Discount per Point*: Bestimmt für wieviele Währungseinheiten ein Treuepunkt eingelöst werden kann.
- *Turnover for Point*: Definiert, wieviel Nettoumsatz zur Vergabe eines Treuepunktes führt.

Bei Eingabe einer neuen Verkaufsrechnung an Kunden müssen die konfigurierten Parameter abgefragt werden, um entsprechend Rechnungsrabatte zu erteilen. Wird die Rechnung danach gebucht, muss zur Behandlung der Treuepunkte in die Buchungslogik eingegriffen werden. Einerseits muss vor dem Buchen geprüft werden, ob der Kunde auch genug verfügbare Treuepunkte hat, um den Rechnungsrabatt mit seinem Punktekonto ausgleichen zu können. Diese zusätzliche Prüfung ist wichtig, da Erfassung und Verbuchung der Rechnung nicht zwangsweise zum selben Zeitpunkt erfolgen müssen. Andererseits muss nach erfolgreichem Verbuchen der Rechnung das Treuepunktekonto des Kunden angepasst werden, dabei müssen sowohl eingelöste, als auch durch die Rechnung verdiente Punkte berücksichtigt werden.

C/AL

AL

### 5.2.3 Datenträgerexport

C/AL

AL

### 5.2.4 Reporting

C/AL

AL

### 5.2.5 Webservice-Anbindung

C/AL

AL

## Kapitel 6

# Tests und Evaluierung

Kapitel 7

Diskussion

# Quellenverzeichnis

## Literatur

- [1] Edward A. Duplaga und Marzie Astani. „Implementing ERP in Manufacturing“. 20 (Juni 2003), S. 68–75. URL: <http://www.dsg.univr.it/documenti/OccorrenzaIn/matdid/matdid965805.pdf> (siehe S. 3).
- [2] Dr Bernard Wong und David Tein. „Critical Success Factors for ERP Projects“ (Jan. 2003), S. 1–2. URL: [https://www.researchgate.net/publication/229022123\\_Critical\\_Success\\_Factors\\_for\\_ERP\\_Projects](https://www.researchgate.net/publication/229022123_Critical_Success_Factors_for_ERP_Projects) (siehe S. 3).
- [3] Mark Brummel. *Learning Dynamics NAV Patterns. Create solutions that are easy to maintain, are quick to upgrade, and follow proven concepts and design*. Packt Publishing, 2015, S. 135–136. URL: <https://www.microsoft.com/en-us/p/learning-dynamics-nav-patterns/fgqpf3h0qc1j?activetab=pivot%3aoverviewtab> (siehe S. 4).
- [4] Michaela Gayer. *Microsoft Dynamics NAV - Einführung in Design und Programmierung*. mbst books, 2016 (siehe S. 7).
- [5] Jürgen Ebert Michaela Gayer Christian Hauptmann. *Microsoft Dynamics NAV 2018: Das Anwenderbuch zur Abwicklung von Geschäftsprozessen*. Carl Hanser Verlag GmbH Co KG, 2018 (siehe S. 5).
- [6] Fiona Fui-Hoon Nah, Silvana Faja und Teuta Cata. „Characteristics of ERP software maintenance: a multiple case study“. *Journal of Software Maintenance and Evolution: Research and Practice* 13.6 (2001), S. 399–414. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.239> (siehe S. 5).
- [7] Kay Giza Tobias Kahlert. *Visual Studio Code Tips and Tricks*. Microsoft Press, 2016. URL: <https://www.microsoft.com/germany/techwiese/aktionen/visual-studio-code-ebook-download.aspx> (siehe S. 15).