

Bitwork iPad app Overview

Basically we need an APP for several mobile devices, but first of all for the iPad 4.

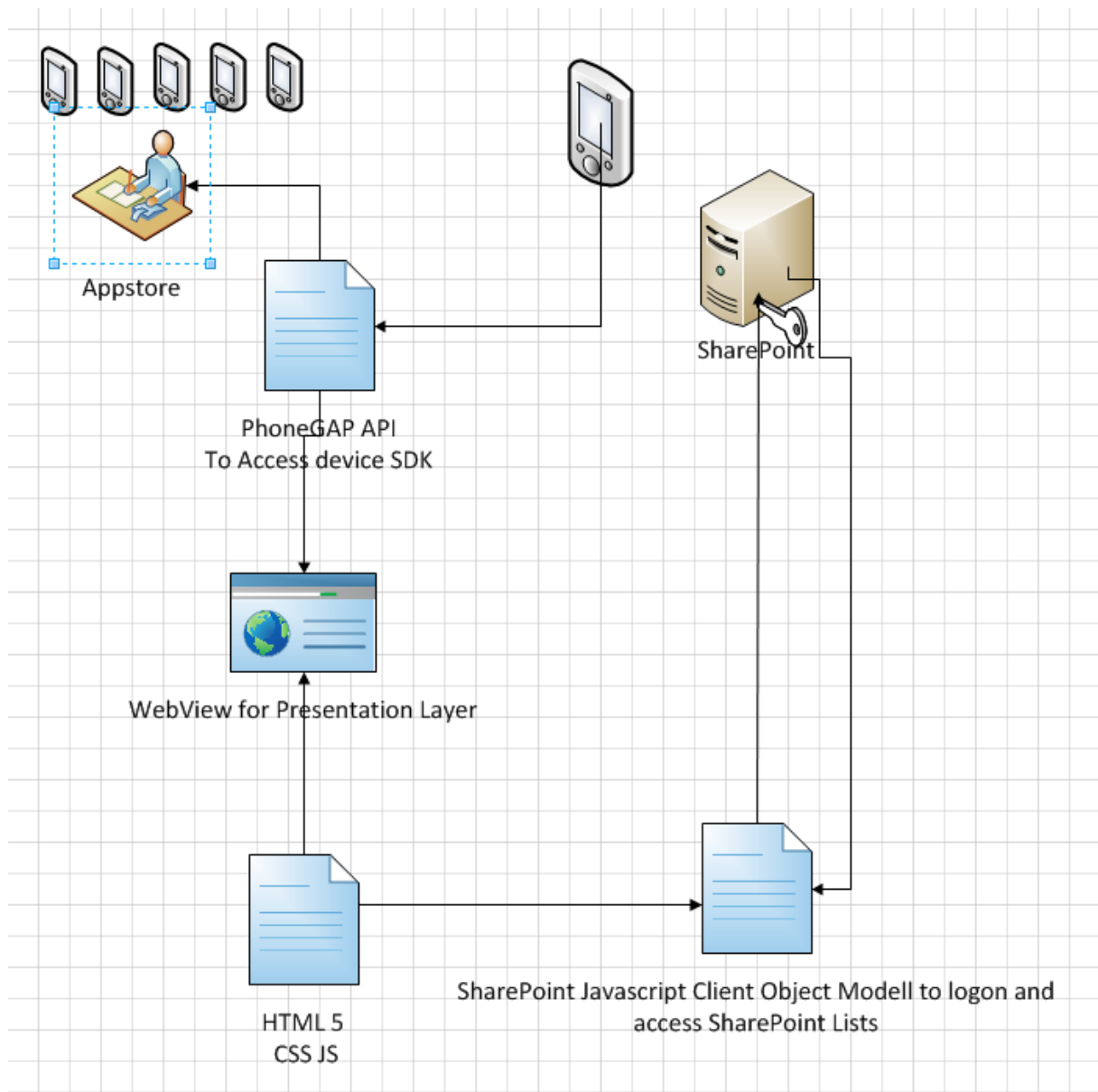
The Application needs to be built with the PhoneGap Application Framework, which is a platform for the cross platform development.

PhoneGap allows us to develop for all possible mobile Platforms based on HTML 5, JS and CSS3.

PhoneGap has a convenient API to access the devices internal features, like filesystem, camera, calender and also a good documentation. PhoneGap also allows us to compile and publish to different appstores automatically. No need to maintain the native SDKs.

It's also possible to use the native API and combine it with a WebView based on HTML 5 for Features which don't work with the PhoneGap API

As Source Control service Github will be used



API Reference		android	blackberry (6)	blackberry10	ios	wp7 (Windows Phone 7)	wp8 (Windows Phone 8)	win8 (Windows 8)	tizen	webos	symbian
Accelerometer	phonegap	✓ Mac, Windows, Linux	X	✓ Mac, Windows	✓ Mac	✓ Windows	✓ Windows	X	X	✓	✓
Camera	CLI										
Capture	Local SDK Support	✓	✓	✓	✓	✓	✓	✓	✓	X	X
Compass	Remote PhoneGap Build	✓	✓	X	✓	✓	X	X	X	✓	✓
Connection	Embedded WebView	✓ (see details)	X	X	✓ (see details)	X	X	X	X	X	X
Contacts	Plug-in Interface	✓ (see details)	✓ (see details)	✓ (see details)	✓ (see details)	✓ (see details)	✓	X	X	X	X
Device											
Events											
File											
Geolocation											
Globalization											
InAppBrowser											
Media											
Notification											
Splashscreen											
Storage											

Platform APIs											
Accelerometer	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Camera	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Capture	✓	✓	✓	✓	✓	✓	✓	X	X	X	X
Compass	✓	X	✓	✓ (3GS+)	✓	✓	✓	✓	✓	✓	X
Connection	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Contacts	✓	✓	✓	✓	✓	✓	✓	✓	X	X	✓
Device	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Events	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
File	✓	✓	✓	✓	(partial)no File Transfer	(partial)no File Transfer	✓	X	X	X	X
Geolocation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Globalization	✓	✓	X	✓	X	✓	X	X	X	X	X
InAppBrowser	✓	✓	✓	✓	✓	✓	X	X	X	X	X
Media	✓	X	✓	✓	✓	✓	✓	✓	X	X	X
Notification	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓



This app will need to connect to a SharePoint server 2010 and possibly 2013.

On this SharePoint are several lists and libraries which the app will need to connect to.

There is supposed to be an offline functionality which synchs some of these lists and libraries to the device. Using XML or JSON into an WebSQL DB.

As the synchronization should only be possible while connected to a wireless network, it shouldn't be a problem to just wipe all local data but the files and download them again. **The Files should only be overwritten if needed. All the files will probably be a few gigabyte of data.**

The offline files need to be findable with a search function.

Attached files of a library need to be stored in the applications sandbox file system and need to be opened from there (files like PDF [uses the devices internal functions to open a pdf or office file])

Additionally there are different tasks the app needs to accomplish:

Appointments added to the devices calendar

Contacts added to the devices contacts list

Local Files needs to be send via email as Attachment

In the logon prompt (form based authentication) has to be an invisible counter, which allows for 3 (7) incorrect logins. After that all local stored information will be deleted.

List of general Tasks:

Connect to SharePoint via JSCOM

Disconnect from SharePoint

Open / Connect to SharePoint List / Library

Iterate SharePoint List / Libraries Elements

Download Metadata of SharePoint List in Web SQL DB or XML File

Download files from Library into local file system / sandbox of iDevice

Create a local WebSQL Database

Delete local WebSQL Database

Delete parts of WebSQL database (eg. Tables)

Execute Queries to local Database

Check whether a user is in a specific SharePoint group

Create a function which builds a local database / xml file, based on the data of a SharePoint list or library

Create a database model

More Specific Tasks

The Application Frontend will be German only and is based on HTML 5. The program code and anything done in the UI by you, should be German or English. Bitwork will localize it when needed. The Design will be done by bitwork GmbH and of course isn't final yet. Also bitwork GmbH will take part in the development process.

The app will be used internally by the customer. It will be published in the B2B Apple Appstore, not in the general Appstore

To sum it up, the App should allow the user to get News and Information about specific products in a so called "machine price list". These news and information can be SharePoint Metadata or documents (mostly pdfs). All the app does is getting information. No information will be send to the SharePoint. Read Only

Login

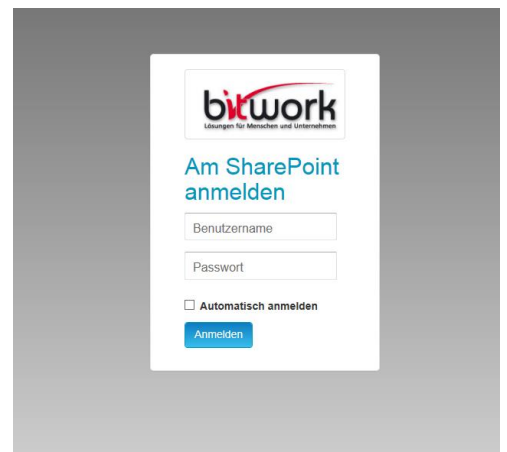
The User needs to log on to the SharePoint.

The SharePoint Adress will be hardcoded.

The Authentication mode is form based authentication.

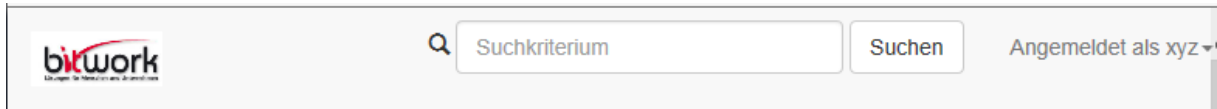
The Customer demands (for what reason we don't know) that after **seven** false logins the app deletes all local files / database entires saved by the app (not the app itself).

The option "automatisch anmelden" means: sign on automatically. So the app should be able to save the login.

The image shows a login form for Bitwork. At the top is the Bitwork logo with the tagline "Lösungen für Menschen und Unternehmen". Below the logo is the text "Am SharePoint anmelden". There are two input fields: "Benutzername" and "Passwort". Below these fields is a checkbox labeled "Automatisch anmelden". At the bottom is a blue button labeled "Anmelden". The form is set against a light gray background.

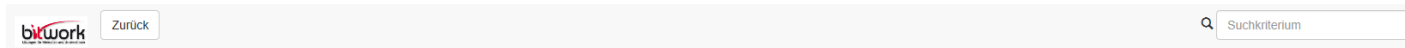
Overall Navigation

On top of all pages but the logon page will be this navigation bar:



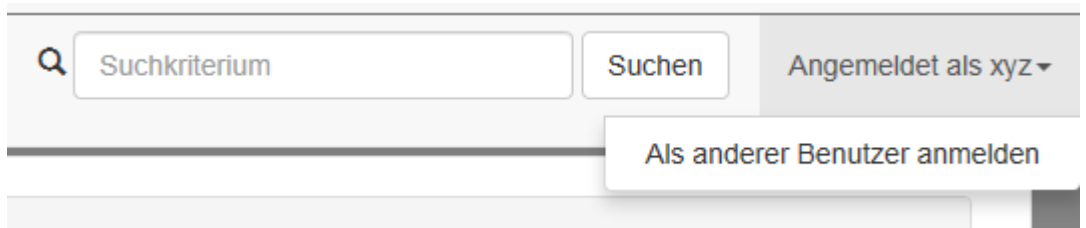
From left to the right:

Brand Logo. By Clicking on the Logo, the user will get back to the main menu. Additionally it is possible to show a backward navigation, which steps one step back:



Next to the navigation is the search box. With this functionality the app should show a record list **grouped** by the type of information (news, calendar entry, phone number, link, information and products, file) based on the entered keyword.

The last Part “angemeldet als” means logged in as. This panel should show the current username. Clicking on it opens a dropdown panel which allows the user to reconnect via the login page (“als anderer Benutzer anmelden”

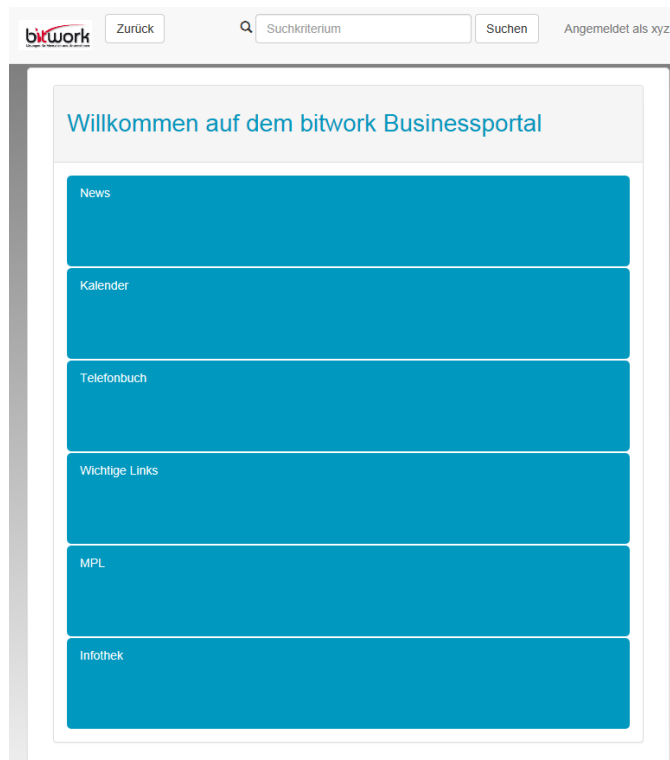


Main menu

The main menu will be a hub to the more specific information available in the app.

So there's in order from top to bottom: news, calendar, phonebook, "important links", a feature called "MPL" (stands for **machine price list**) and "Infothek" which is a list of additional files not specific to a product in the "MPL".

This means there will be documents linked to a specific product (in the MPL) and none product relevant documents (Infothek).



News

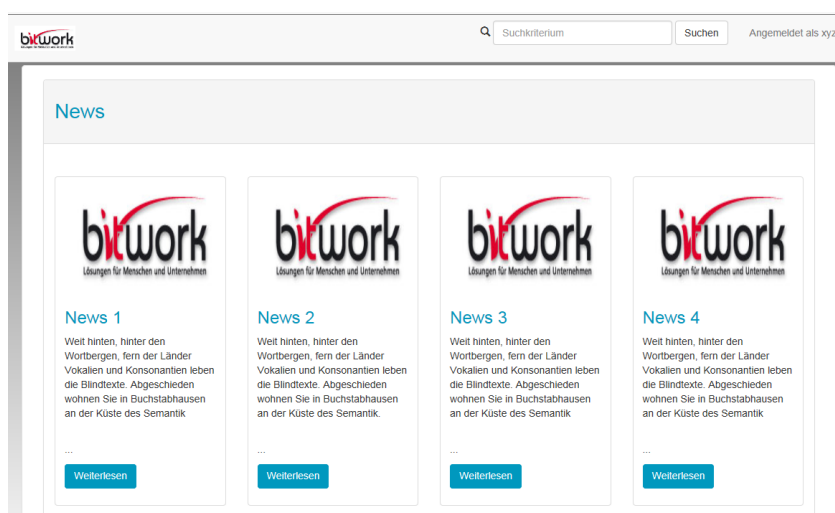
The News Page needs to aggregate data from a SharePoint list called "news".

The News function does not need to be available while offline.

So far the List only contains a title column and a news column (multiline rich text + documents), an expiration date. If it causes trouble (We have never read from a rich text multiline box yet), we could probably change the design (for example Title, News, Picture, File column).



This list needs to be displayed in the app [maybe the text needs to be shortened for layout purposes]. A click on "weiterlesen" will open the browser and show the news in SharePoint.



Its not necessary to show the rich text or the document in the App.

Calendar

The calendar is similar to the news site. Not all information (columns) need to be shown. It's sufficient to show the Title, Location, Start and End date ("Endzeit"). News which end date is expired should not be shown.

The News function does not need to be available while offline. It would be nice (if the Phonegap API allows it) to add the entry to the calendar of the phone/tablet.

Anfangszeit	Datum und Uhrzeit	Ereignis
Auf doppelte Einplanung überprüfen	Auf doppelte Einplanung überprüfen	
Beschreibung	Mehrere Textzeilen	Ereignis
Beschreibung3	Mehrere Textzeilen	Ereignis
Endzeit	Datum und Uhrzeit	Ereignis
Frei/Gebucht	Frei/Gebucht	
Kategorie	Auswahl	Ereignis
Ort	Eine Textzeile	Ereignis
Ressourcen	Ressourcen	
Sichtbar für Login Admin	Ja/Nein	Ereignis
Sichtbar für Login Händler	Ja/Nein	Ereignis
Sichtbar für Login Intern	Ja/Nein	Ereignis
Sichtbar für Login Kunden(Ing. Büros)	Ja/Nein	Ereignis
Teilnehmer	Person oder Gruppe	
Titel	Eine Textzeile	Ereignis
Erstellt von	Person oder Gruppe	
Geändert von	Person oder Gruppe	

Handelsschulung: Ölgeschmarte Kollben- und Schraubenkompressoren

Handelsschulung Ölgeschmarte Kollben- und Schraubenkompressoren

Zeitplan:

Seminarschulung der Atlas Copco Vertragshändler. Anweisung durch den Vertragshändler, max. 10 Teilnehmer.

Dauer der Schulung: 2 Tage


Kosten / Preisniveau: 280,00 Euro

Kontakt für organisatorische Fragen:
Frau Daght Wenz, CTO-Essen (+49 201 2177 383, twenz@atlas-copco.com)

 [2012-1107_HSK_Handelschulung Ölgeschmarte Kollben- und Schraubenkompressoren_HK.pdf](#)

Trainingscenter Essen
Langerhakenstraße 10, 43141
Essen


04.11.2013 08:30 04.11.2013 17:30



Suchkriterium

SuchenAngemeldet als xyz

Kalender



Lösungen für Menschen und Unternehmen


Mastering of the Academy

08.11.2013 08:30 - 17:30 Uhr

08.11.2013

Mastering of the Academy - Schulung für die als Vertragshändler von Atlas Copco Kompressoren und Drucklufttechnik verkaufte Schulung in 3 Modulen am 12. + 13. September + 08. + 09. Oktober und 08. + 09. November 2013 Raum Nürnberg, Anmeldebuchung 12. August 2013 Mindestteilnehmerzahl: 10 Personen. Maximale Teilnehmerzahl: 16 Personen.

[Weiterlesen](#)



Phonebook

The phone book is a bit more complicated. The Phonebook is based on a picture library in SharePoint.

Inside the picture library are folders, which will be used to build a hierarchy. Inside the folders lies the pictures.


For example:


Root folder

- Department 1
 - user x
 - Sub department 1
 - user y
- Department 2
 - user w

Inside the app the user needs to navigate through the folders in a tree view. If he clicks on a folder its contents will be displayed. If he clicks on an entry the detail information need to be loaded. The tree view needs to load the folders hierarchically. To save this a file structure/xml/database model needs to be created with a parent folder relationship. Additionally to the global search function, this page needs a phonebook only search bar. The search takes names and finds them.

It would be nice (if the Phonegap API allows it) to add a user of the phonebook to the phones contact list.

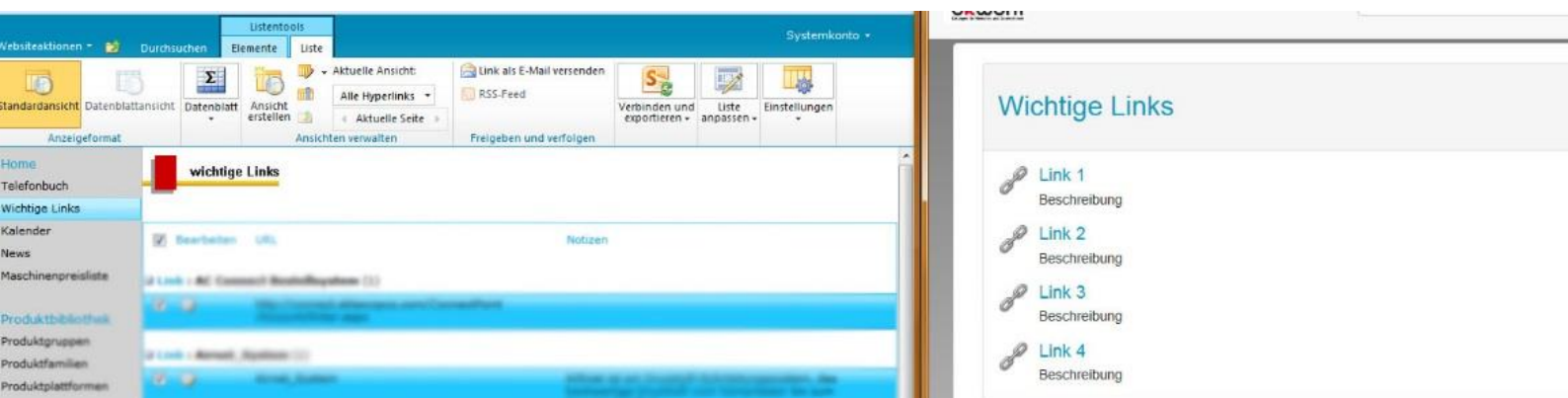




Vorname:	Joel
Nachname:	Gump
Telefon:	+49 205 33 77 100
Mobilfunknummer:	
Fax:	
E-Mail:	joel.gump@de.affinity.com
Abteilung:	Geschäftsführung
Funktion:	Geschäftsführer
Stellenbeschreibung:	
Vertreter:	Kein Vertreter

Important Links

Important Links is based on a SharePoint Link list. All it does is present a hyperlink (Url) and a description field ("Notizen"). These links need to be stored locally



Machine Price List

Next to the phone book the machine price list will be the hardest feature to archive. The customer has a **horrible** categorization "technique" which we need to adapt to. This topic will probably need some discussion, but we will try to list it up:

We have several Lists:

Product groups

Product families

Product platforms

Product

Product Description "Other"

Product Description "Equipment"

Product Description "Options"

Product groups (**1 Group has n families** and also **one family can be in n groups**)

-- Product families (1 Family has n platforms)

--Produkt platforms (1 Platform has n Products)

--Products (Products has either n
Equipment's or n Other Products)

--Product Equipment

-- Product Option

--Product Other

... So far so good

A Product Option can either belong to a Product Group, Product family, a Platform, a Product, or an Equipment.

The goal for the MPL is a list in which the user sees all Product groups. Once he clicks on a group, all Families which belong to the Product group are loaded. The user then clicks on a family, which will open the platforms. And so on. If he reaches a product Equipment, the Equipment including possible Product Options must be loaded and shown in detail. If he reaches an "other Product" its details must be loaded from the "Product description other" list.

To make things even worse there's documents in a document library which are linked to either a group, family, platform, product, equipment or other. So additionally to the metadata, there can be documents.

For example, if there is a Family called **Family1** and a product **Product1** and a document which belongs to **family1** it needs to be shown on the detail page of **Product1** grouped by the Documents "Documenttype".

In the DEMO Vmware there is a webpart which does the same. But probably **bitwork will do this code part, as we have already done it in the past.**

Productgroup:

Single Column: Productgroup (internal name Title)

Productfamily:

Column: Productfamily (internal name Title) type Text

Column: Productgroup: Lookup Value **required**

Productplattform:

Column: Productplattform (internal name Title) type Text

Column: Productfamily: Lookup Value **required**

Product:

Column: Product (internal name Title) type Text

Column: Productplattform: Lookup Value **required**

Product Equipment:

Column: Product Equipment (internal name Title) type Text

Column: Product: Lookup Value **required**

Additional Metadata to the Equipment (like ID, Price eg.)

Product Others:

Column: Product Others (internal name Title) type Text

Column: Product: Lookup Value **required**

Additional Metadata to the Other Product (like ID, Price eg.)

Product Options

Column: Product Options (internal name Title) type Text

Additional Metadata to the Product Option (like ID, Price eg.)

Column: Product Equipment: Lookup Value **NOT REQUIRED**

Column: Product: Lookup Value **NOT REQUIRED**

Column: Product Platform: Lookup Value **NOT REQUIRED**

Column: Product Family: Lookup Value **NOT REQUIRED**

Column: Product Group: Lookup Value **NOT REQUIRED**

Documents:

Column: Name (internal name Title) type Text

Column: Document Type: Lookup Value **REQUIRED**

Column: Product Equipment: Lookup Value **NOT REQUIRED**

Column: Product: Lookup Value **NOT REQUIRED**

Column: Product Platform: Lookup Value **NOT REQUIRED**

Column: Product Family: Lookup Value **NOT REQUIRED**

Column: Product Group: Lookup Value **NOT REQUIRED**

Dokumenttyp	Typ	Name	Produktgruppe	Produktfamilie	Produktplattform	Produkt	Equipment
Technische Beschreibungen		1-9_1_21_01_20 - 201_01_20 - 201_01_20 -	Verfahren	01v 20 10a 200			
Technische Datenblätter		1-9_1_21_01_20 - 201_01_20 - 201_01_20 -	Verfahren	01v 20 10a 200			

Abbildung 1 Documents to products

Product Information

Produktdetailinformationen	
▼	Produktdetailinformationen
Produktbezeichnung	Erweiterter KSP Q=5
Teilenummer	280000000
Listenpreis	26,10

Dokumente	
▼	Angebotstext
<ul style="list-style-type: none"> 2012 1030 KSP_Affnet technische Eigenschaften Angebot_IF.docx 2012 1030 KSP_Affnet Vorteile Angebot_IF.docx 	
▼	Betriebsanleitungen
Affnet Montagehinweise.pdf	
▼	Kalkulationsprogramme
<ul style="list-style-type: none"> Affnet Pressure Calculator 247.exe.zip Affnet H50_2012_willkommenring_release.xlsm Affnet Quoter 247.xlsm Affnet Quoter 247 User Reference.pptx 	
▼	Produktpräsentation
<ul style="list-style-type: none"> 2012 1030 KSP_Affnet Planner 1_IF.jpg 2012 1030 KSP_Affnet Filterinstallation_IF.jpg 2012 1030 KSP_Affnet Schulung_IF.pdf 	
▼	Prospekte
<ul style="list-style-type: none"> 2012 1030 KSP_Affnet Broschüre_IF.pdf 2012 1030 KSP_Affnet Training_IF.pdf 	
▼	Zertifikate
2012 1030 KSP_Affnet Free_IF.pdf	

Product Option to the Product

Name	Optionenbezeichnung	Teilenummer	Listenpreis
No records to display.			

Abbildung 2Detail page of a Product Equipment, Product Option or a "other Product"

Infothek

Infothek is easy. Infothek is a document library which is has an item level permission management. Every user automatically sees what he is allowed to see. There are no specific metadata in this library. So basically all we need is to dynamically save the folder structure with its files and present it to the user in a tree view. These files will be PDF files. The Document management will be handled by the Device. If the user lacks a PDF/Office Reader, it's the user's problem, not ours.

Search

As stated before, the search should aggregate all the local stored information into a result list. The search should work as a %like% operation. It only needs to look for the corresponding Title (not all metadata)

The Groups would be:

- Product group
- Product family
- Product Platform
- Product
- Product Equipment
- Product Option
- Product Other

Phonebook Name

Infothek Document Title

Product Document Title

Clicking on one of these Elements is supposed to open the entry

Database Model

Tables:

Registration

Purpose: Store Offline Login information

Fields: ID (Required), Domain, Loginname, Password (should be crypted [For Example SHA1])

SharePoint List "News"

Purpose: Store News. Can be deleted after Expiration Date

Fields: ID (Required), Title, Body (much text), ExpirationDate (Date), Picture_FK

SharePoint List "Calendar (Kalender)"

Purpose: Store Events. Can be deleted after Expiration Date

Fields: ID (Required), Title, Body (much text), StartDate (Date), Location, ExpirationDate (Date), Picture_FK

Pictures (Bilder)

Purpose: Store News and Calendar pictures. Either path to Sandboxfile or BLOB.

So Far, these do not exist in SharePoint.

Version A (Blob): ID (Required), Title, File (byte)

Version B (Path): ID (Required), Title, FilePath

SharePoint Table "ImportantLinks" (WichtigeLinks)

Purpose: Store Hyperlinks with a description

Fields: ID, Link, Description

SharePoint Table "Phonebook" (Telefonbuch)

Purpose: Store Contact Data in a folder hierarchy. Can be either Folder or File(Picture)

Version A (Blob): ID (Required), Name, Forename, Phone, MobilePhone, Fax, Email, Department, Function(Text), IsFile(Boolean), ParentFolder , Picture(Blob)

Version B (Path): ID (Required), Name, Forename, Phone, MobilePhone, Fax, Email, Department, Function(Text), IsFile(Boolean), ParentFolder , PicturePath

SharePoint Table "Product Groups (Produktgruppen)"

Purpose: MPL Hierarchy

Fields: ID (Required), Productgroup (Required)

SharePoint Table "Product Families (Produktfamilien)"

Fields: ID(Required), Productfamily(Required), Productgroup_FK(Required)

SharePoint Table "Product Product Platform (Produktplattformen)"

Fields: ID(Required), Productplattform(Required), Productfamily_FK(Required)

SharePoint Table "Product (Produkt)"

Fields : ID(Required), Product (Required), Productplattform_FK(Required)

SharePoint Table "Equipment Product" (Produktbezeichnung Equipment)"

Fields: ID(Required), ProductEquipment (Required), Product_FK(Required), Variation, Piecenummer,maxPressure, FAD, kW, Price, Cooling

SharePoint Table "Other Product" (Produktbezeichnung Sonstige)"

Fields: ID(Required), OtherProduct (Required), Product_FK(Required), Piecenummer, Price

SharePoint Table "Product option" (Produktbezeichnung Optionen)"

Fields: ID(Required), ProductOption (Required), Product_FK, ProductGroup_FK, ProductFamily_FK, ProductPlattform_FK, EquipmentProduct_FK , Variation, Piecenummer,Price

SharePoint Table "Documenttypes" (Dokumenttypen)"

Purpose: Lookuptable for Documents

Fields: ID (Required), Title (Required)

SharePoint Table "ProductDocuments" (Dokumente)"

Purpose: Documents for a specific Productgroup, Family, Platform, eg

Fields: ID (required), DocumentType_FK, ProductGroup_FK, ProductFamily_FK, ProductPlatform_FK, Product_FK, EquipmentProduct_FK, OtherProduct_FK, ProductOption_FK

SharePoint Table "OtherDocuments" (Infothekhaendler)"

Purpose: general Documents, not bound to a product group, family, eg. Contains Files and Folders

Fields: ID (Required), Title (Required), ParentFolderID, IsFile(Boolean [if not it's a folder])

Version Control:
www.github.com

After a free registration, bitwork will invite you to our github repository

Github can be used via SVN

~~Einzelne Funktionen~~

~~Anmelden am SharePoint~~

~~Abmelden vom SharePoint~~

~~Öffnen einer SharePoint Liste~~

~~Öffnen und iterieren einer SharePoint Bibliothek~~

~~Herunterladen der Metadaten einer Liste & Bibliothek in WEB SQL DB oder XML~~

~~Herunterladen der Dateien einer Bibliothek in lokales Dateisystem vom Gerät~~

~~Löschen der internen Datenbank~~

~~Löschen von Teilen der internen Datenbank~~

~~Anlegen der lokalen Datenbank~~

~~Füllen der lokalen Datenbank~~

~~Funktion die aus den Informationen (Metadaten) einer SharePoint Liste / Bibliothek automatisch eine Datenbanktabelle anlegt. Datentypen sind eher unwichtig, kann alles Text sein.~~

~~Wie loggen wir uns an, wenn der SharePoint Offline ist? Speichern, dass das kennwort korrekt war?~~

Datenbankmodell von uns

One Table for the filelinks, filecategory (news, kalender eg)