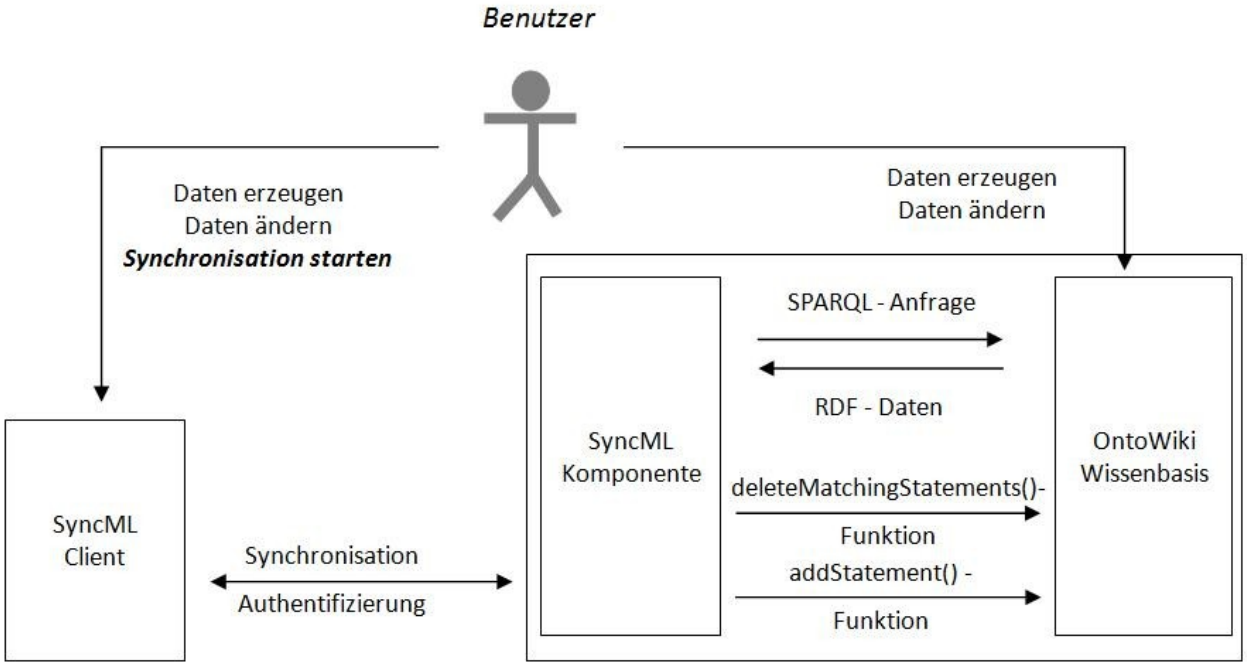
Entwurfsbeschreibung

1.Allgemeines

Die SyncML Gruppe hat sich die Aufgabe gestellt eine Erweiterung für das OntoWiki zu schreiben. Diese Erweiterung soll mobile Clients mit dem OntoWiki verbinden. Besonderes Augenmerk wird auf Kontaktdaten des mobilen Clients gelegt. Diese PIM-Daten des mobilen Clients werden mit dem OntoWiki synchronisiert. Hierbei ist wichtig das nur neu erzeugte oder geänderte Datensätze synchronisiert werden. Bestehende/nicht geänderte Daten werden von der Synchronisation nicht erfasst, da diese bereits auf beiden Seiten auf dem aktuellsten Stand sind.

2.Produktübersicht

Der registrierte Nutzer kann von seinem mobilen Client aus Daten erzeugen und ändern. Auf eigenem Wunsch kann er nur, mithilfe seiner OntoWiki Zugangsdaten, eine Synchronisation auf dem Client anstoßen. Bei einer richtigen Authentifizierung werden die veränderten Datensätze verglichen. Diese Daten werden mittels XML oder WBXML gesendet und vom OntoWiki interpretiert. Eine einfache Darstellung der Produktübersicht finden sie in der folgenden Grafik.

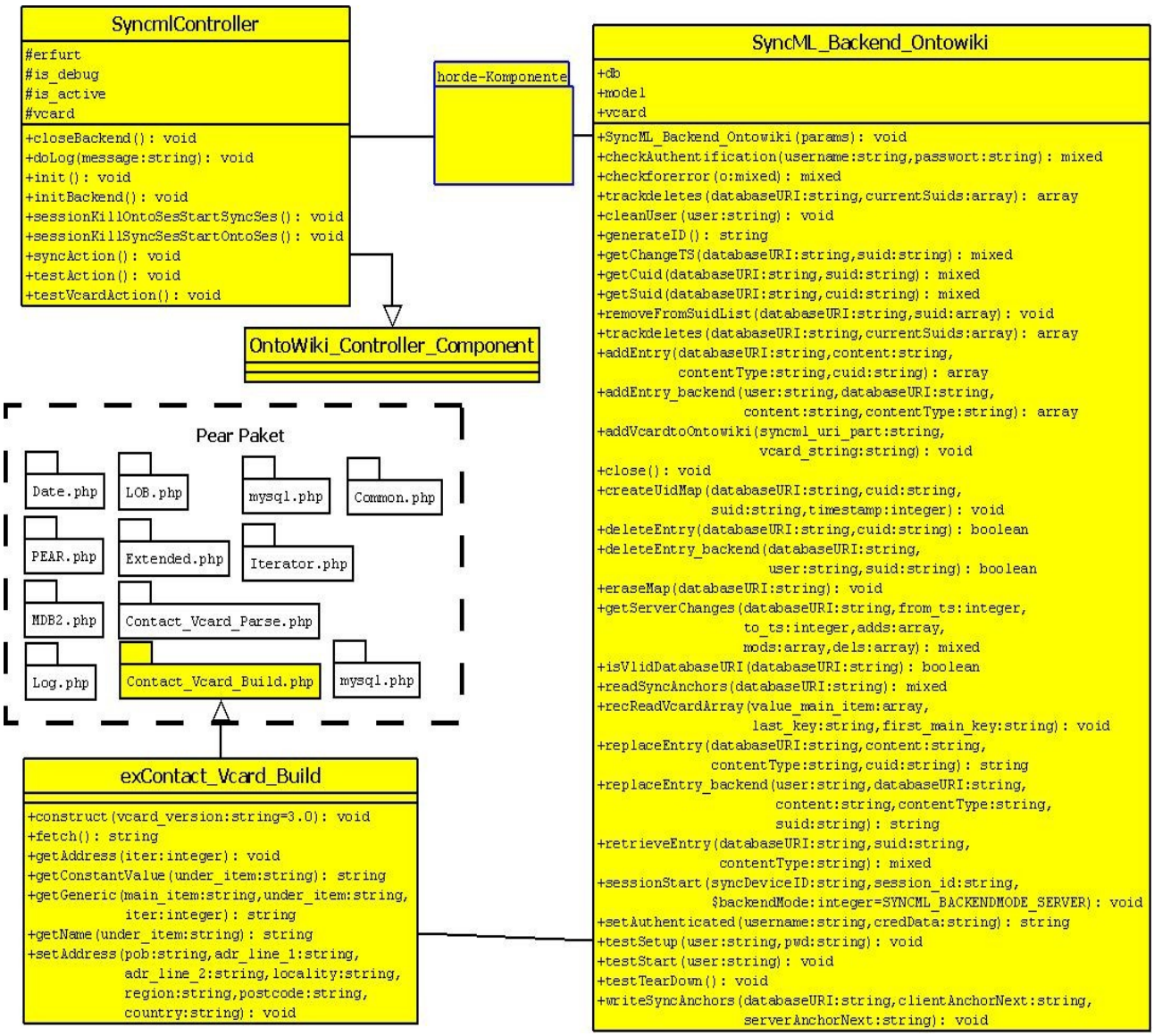


3.Grundsätzliche Struktur- und Entwurfsprinzipien des Gesamtsystems

3.1Allgemeine Beschreibung und statisches Modell

Das OntoWiki basiert auf einer MVC Struktur, die in unserer SyncML-Komponente nur teilweise umgesetzt werden kann. Grund dafür ist dafür der fehlende View, der dem User eine graphische Oberfläche bieten sollte. In unserem Projekt sollen ausschließlich im Hintergrund die Daten synchronisiert werden.

Die Klasse OntoWiki ist unser Backend.Sie steuert die ganzen Vorgänge der Synchronisation zwischen Client und OntoWiki. Die ganze Abfrage findet in ihr statt und auch der Datenaustausch wird von ihr gehandhabt.
Der SyncMLController ist bestandteil von OntoWiki und erbt von deren KomponentenController Klasse.Dadurch ist diese Klasse die Anlaufstelle für die Synchronisation zwischen dem Client und dem OntoWiki.
Die letzte von uns bearbeitete Klasse ist exContact_Vcard_Build und wird benötigt um eine Vcard zu ändern oder zu konvertieren.



3.2 Dynamisches Modell und Szenarien

3.2.1 Szenarien

1. Story

Status: abgeschlossen

OntoWiki Komponente welche den SyncML-Server enthält und keinerlei eigene Funktionen bereitstellt. Keine Änderungen am SyncML-Server. Komplette Arbeit auf der MySQL-Tabelle.

2. Story

Status: abgeschlossen
Personen: Felix Liebmann

OntoWiki Komponente wird hier noch nicht verändert. Die Authentifizierung übernimmt nun das Backend(OntoWiki), was bedeutet, dass dies bei dem Backend(SQL) weg fällt und somit auch die Tabelle syncml_suid bei der MySQL-Tabelle nicht mehr benötigt wird.

Die Authentifizierung besteht aus:

- Abfrage ob Username, Passwort Kombination besteht erledigt
- Prüfen ob User ein bestimmtes Model besitzt erledigt
- Prüfen ob User Schreib- und Leserechte auf diesem Model besitzt erledigt
- Prüfen ob der User die Action Sync ausführen darf ausstehend

Eine erfolgreiche Authentifizierung wird mit Usernamen quittiert.
Eine Authentifizierung ist erfolgreich wenn alle 4 Bestandteile erfolgreich abliefen.
Eine missglückte Authentifizierung wird mit einem false quittiert.

3. Story

Status: abgeschlossen
Personen: Konrad Abicht
Uwe Seiler
Katja Künne

Der Server besitzt 3 Änderungsfunktionen für die Datenbank, diese wären: add, delete und change. Wenn diese Funktionen ausgeführt werden, wird nicht nur die MySQL Datenbank, sondern auch die OntoWiki Datenbank inhaltlich verändert. Dies wird durch einen Aufruf von einer von 3 Funktionen bewerkstelligt. Man brauch sich nicht darum zu kümmern, welche Art der Änderung man durchzuführen hat, da der SyncML Server selbst entscheidet ob gerade ein bestimmter Datensatz angelegt, gelöscht oder verändert wird.

Anlegen Funktion:

- Aufruf von addStatement (URI(Model des Users),
  Subjekt(syncmlid),
  Prädikat("has"),
  Objekt(vCard))

Löschen Funktion:

- Aufruf von deleteMatchingStatements(URI(Model des Users),
  Subjekt(syncmlid),
  Prädikat("has"),
  Objekt(vCard),
  ModelIri(?))

Änder Funktion:

- Aufruf der Lösch-Funktion und danach der Anlege-Funktion
-

4. Story

Status: abgeschlossen
Personen: alle

Nach erfolgreicher Authentifizierung wird eine neue Funktion namens SyncOntoWikiWithSyncml() aufgerufen und es werden alle Daten aus dem Model des Users in die SyncML-Tabellen gespeichert. Dabei werden alle vorhandenen Daten in der SyncML-Tabelle syncml_data gelöscht, eventuelle Verknüpfungen in der Tabelle syncml_map ebenfalls.

Synchronisierungsfunktion:

- Aufruf eines DELETE FROM Statements auf Tabelle syncml_data um die Daten des Users zu löschen
- Einfügen des kompletten Datenbestandes aus dem Model in die syncml_data Tabelle mittels INSERT INTO Statements
- Abfragen aller Subjekte, Prädikate und Objekte aus dem Model des Users
- Generierung eines Arrays
- Iterierung über dieses Array und Einfügen der Datensätze

(bei Anstrich 1+2 wird direkt SQL genutzt)

5. Story

Status: ausstehend
Personen: Konrad Abicht
Lars Eidam
Uwe Seiler

Hier werden wir nun dafür sorgen, dass die VCards in RDF umgewandelt werden und so direkt in die OntoWiki-Datenbank gespeichert werden. Diese Story ist somit nur eine Verbesserung der 3. Story, denn dort wurden die VCards lediglich als String behandelt und dies soll hier geändert werden.
Um dies zu ermöglichen erstellen wir der Übersicht halber eine neue Klasse namens Vcard.php. In dieser wird der String, der die VCard beinhaltet, mittels der Funktion readVcard() untersucht und in ein Array geparst, damit dieser dann später bearbeitet werden kann., Diese Array bestehht aus zwei Dimensionen, welches in der ersten Dimension die VCard-Attribute besitzt (z.B. Tel;Work, Tel;Home, FN oder N) und die zweite Dimension den eigentlichen Attribut-Wert. Nach Abschluss dieser Konvertierung werden die Elemente in der OntoWiki-Datenbasis mit Hilfe der Funktion „fromVcardToOntowiki()“ ein gepflegt und wird somit folgende Formation unterliegen:

z.B.

Subjekt:	*syncml: 20090618194308.48576i8l2led1vs4@localhost*
Prädikat:	http://www.w3.org/..../3.0#N
Objekt:	„Hans“

6. Story

Status: ausstehend
Personen: Konrad Abicht
Lars Eidam
Uwe Seiler

Auch diese Story basiert auf der zu vorgehenden Überlegung der Story 4. Bei möglichen Änderung oder Löschungen von Daten direkt im OntoWiki muss der Client bei der nächsten Synchronisation informiert werden. Dazu werden die geänderten RDF- Daten aus der OntoWiki-Datenbank mittels „fromOntowikiToVcard()“- Funktion ausgelesen, werden wieder zusammen gefasst und in die

SyncML-Datenbank eingefügt. Ergebnis dieser Funktion ist eine zum Client kompatible VCard mit dem Datentyp String. Diese VCard kann im Anschluss zum Client gesendet werden.

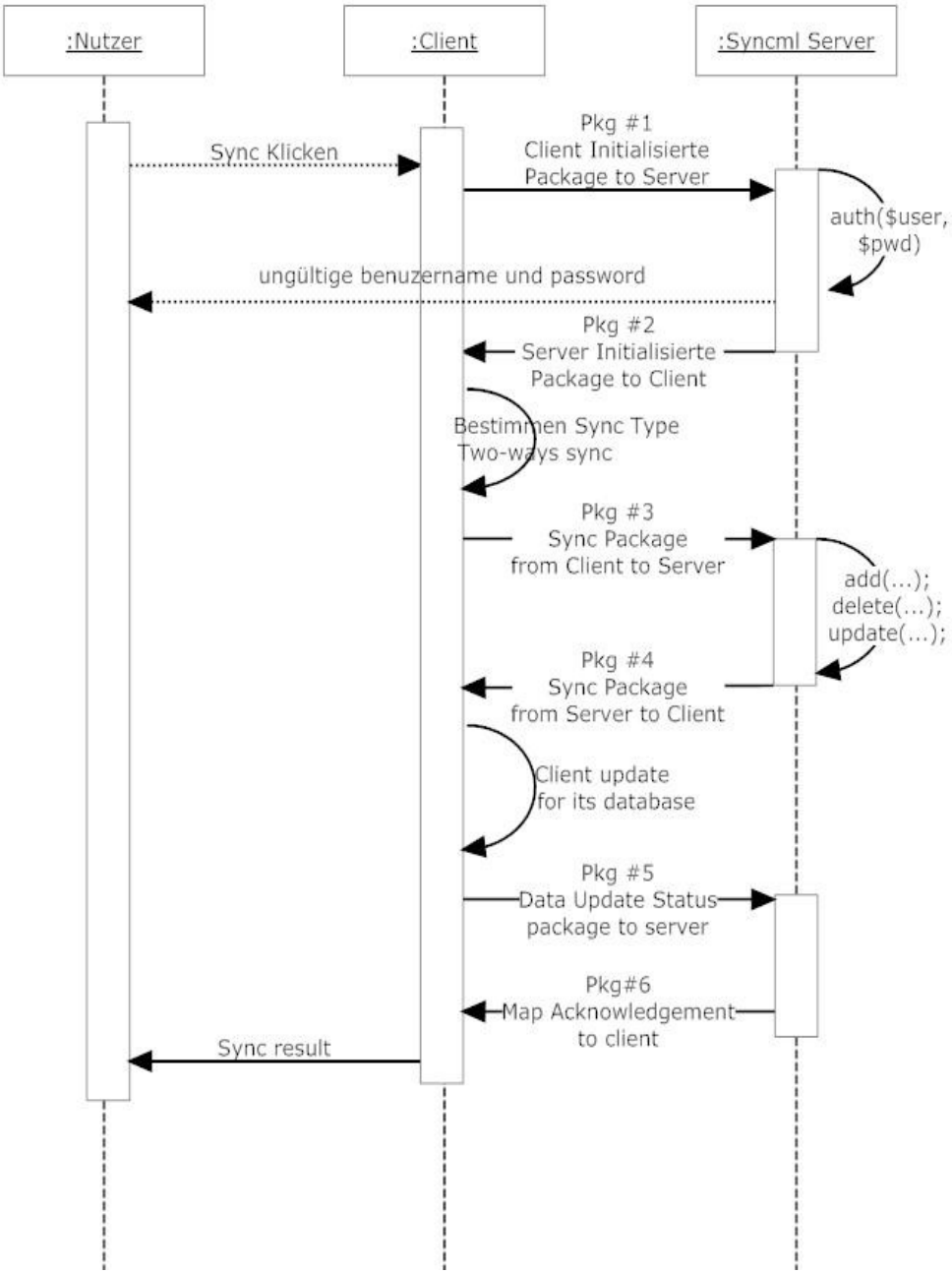
3.2.1 Sequenzdiagramm

1. Sync Initialisierung Phase
   - Ausführung der Authentifizierung zwischen Client und SyncML Server.(Pkg #1)
   - Bestimmung der Protocol der Synchronisationstype und die synchronisierte Database.
   - Es gibt sieben verschiedene Sync-Typen
     - Two-way-Sync
     - Slow-Sync
     - One-way-Sync from client only
     - Refresh sync from client only
     - One-way -Sync from server only
     - Refresh sync from server only
     - Server Alerted Sync

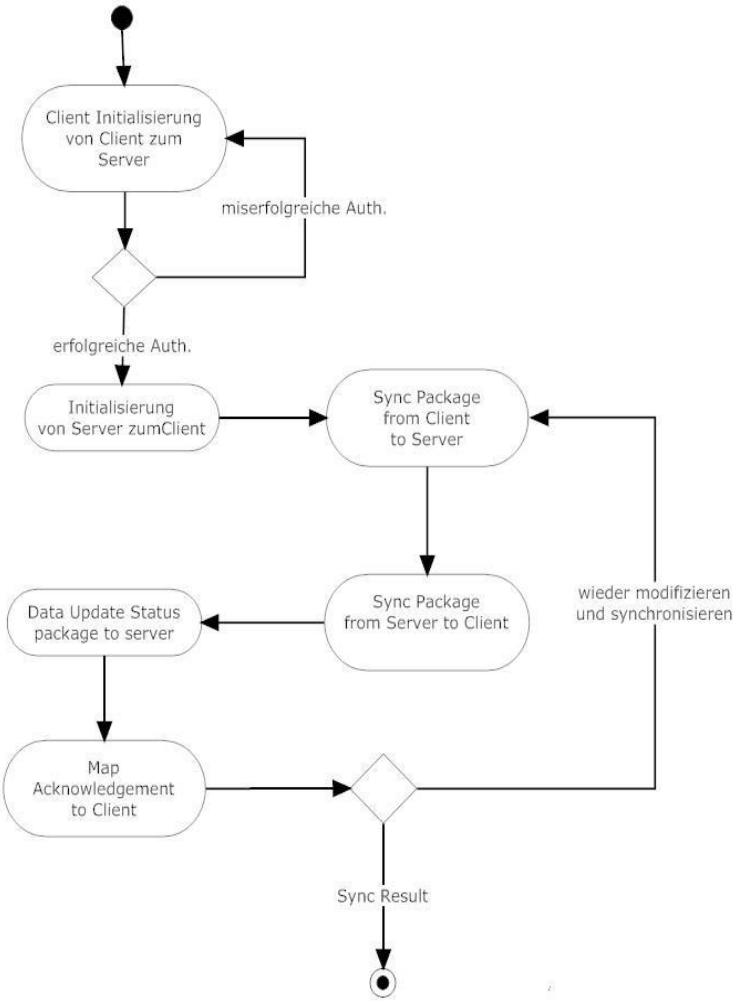
   -Aktivierung des Austausch zwischen Service und Device

2. Sync Phase
   - Synchronisation-Type als Two-Way Sync.
   - zuerst schickt der Client die Modifikation
   - dann bearbeitet die Server die Daten, die von Client geschickt werden

   --> d.h. Aktualisierung der Daten, die am Server Seite gespeichert werden
   - anschließende schickt der Server die Modifikation zum Client
     - Pkg #3: Client Modifikation zum Server.
     - Pkg #4: Server Modifikation zum Client.
     - Pkg #5: Data Update Status von Client. Transformiert die Information über das Ergebnis, dass am Client Seite geändert
     - Pkg #6: Map Acknoledgement vom Server. Antwort des Client, dass der Server die mapping Information aus dem Client erhalten hat

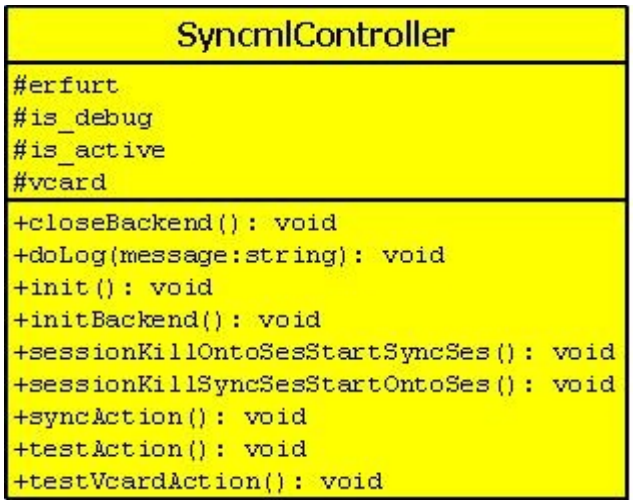

:Nutzer
:Client
:Syncml Server
Sync Klicken
Pkg #1
Client Initialisierte
Package to Server
auth($user,
$pwd)
ungültige benuzername und password
Pkg #2
Server Initialisierte
Package to Client
Bestimmen Sync Type
Two-ways sync
Pkg #3
Sync Package
from Client to Server
add(...);
delete(...);
update(...);
Pkg #4
Sync Package
from Server to Client
Client update
for its database
Pkg #5
Data Update Status
package to server
Pkg#6
Map Acknowledgement
to client
Sync result

3.2.2 Datenflussdiagramm



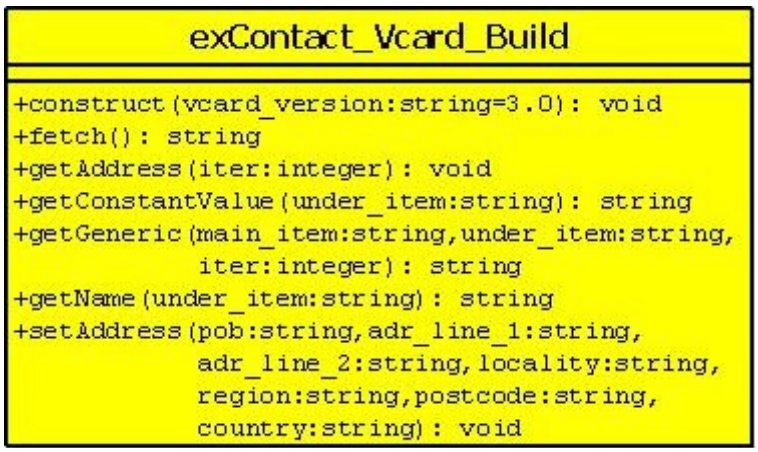
4.Grundsätzliche Struktur- und Entwurfsprinzipien der einzelnen Klassen

SyncmlController
Diese Klasse steuert die Abläufe. Für die Synchronisation muss die OntoWiki Session beendet werden und die Synchronisationssession gestartet werden und danach wieder geschlossen und die OntoWiki-Session aufgebaut werden, dies geschieht hier. Auch wird hier das SyncML Backend initialisiert und später wieder geschlossen. Auch wird hier natürlich die Synchronisation angestoßen und zusätzlich eine Log der Session erstellt.



Horde-Server
Den Horde-Server haben wir in unserem Klassendiagramm nicht weiter betrachtet, da es keine erheblichen Veränderungen in seiner Struktur gegeben hat. Wir verwenden allerdings, anstatt den beiden vorgegeben Backends, unser Klassenkonstrukt um den Horde-Server in das OntoWiki zu integrieren.

exContact_Vcard_Build
Diese Klasse ist eine Erweiterung der Klasse Contact_Vcard_Build und somit erbt sie von dieser. Sie ist für die Konvertierung der vCard zuständig und hier können auch Werte einzelner Attribute abgefragt werden.



SyncML_Backend_OntoWiki

Diese Klasse ist vom SyncML_Backend abgeleitet und erweitert dessen Funktionen um die von uns benötigten. Sie besitzt die Funktion checkAuthentification(), die die komplette Authentifizierung durchführt. Hinzu kommt, das die Klasse auch den kompletten Datenaustausch vornimmt.
So kann die Horde-Komponente von alleine erkennen, ob die Daten bei der Synchronisation hinzugefügt, geändert oder gelöscht werden müssen. Wir haben diese Funktionen nur so verändert, dass sie auch den Datenbestand im Store verändern.

SyncML_Backend_Ontowiki

```
+db
+model
+vcard
```

```
+SyncML_Backend_Ontowiki(params): void
+checkAuthentification(username:string,passwort:string): mixed
+checkforerror(o:mixed): mixed
+trackdeletes(databaseURI:string,currentSuids:array): array
+cleanUser(user:string): void
+generateID(): string
+getChangeTS(databaseURI:string,suid:string): mixed
+getCuid(databaseURI:string,suid:string): mixed
+getSuid(databaseURI:string,cuid:string): mixed
+removeFromSuidList(databaseURI:string,suid:array): void
+trackdeletes(databaseURI:string,currentSuids:array): array
+addEntry(databaseURI:string,content:string,
          contentType:string,cuid:string): array
+addEntry_backend(user:string,databaseURI:string,
                  content:string,contentType:string): array
+addVcardtoOntowiki(syncml_uri_part:string,
                    vcard_string:string): void
+close(): void
+createUidMap(databaseURI:string,cuid:string,
              suid:string,timestamp:integer): void
+deleteEntry(databaseURI:string,cuid:string): boolean
+deleteEntry_backend(databaseURI:string,
                     user:string,suid:string): boolean
+eraseMap(databaseURI:string): void
+getServerChanges(databaseURI:string,from_ts:integer,
                  to_ts:integer,adds:array,
                  mods:array,dels:array): mixed
+isVlidDatabaseURI(databaseURI:string): boolean
+readSyncAnchors(databaseURI:string): mixed
+recReadVcardArray(value_main_item:array,
                   last_key:string,first_main_key:string): void
+replaceEntry(databaseURI:string,content:string,
              contentType:string,cuid:string): string
+replaceEntry_backend(user:string,databaseURI:string,
                      content:string,contentType:string,
                      suid:string): string
+retrieveEntry(databaseURI:string,suid:string,
               contentType:string): mixed
+sessionStart(syncDeviceID:string,session_id:string,
              $backendMode:integer=SYNCML_BACKENDMODE_SERVER): void
+setAuthenticated(username:string,credData:string): string
+testSetup(user:string,pwd:string): void
+testStart(user:string): void
+testTearDown(): void
+writeSyncAnchors(databaseURI:string,clientAnchorNext:string,
                  serverAnchorNext:string): void
```