Bericht über die Exportierbarkeit des spanischen e-APP-Systems

Justizministerium

Mai 2011

INHALTSVERZEICHNIS

1. EINFÜHRUNG UND ZIELE

Das System der elektronischen Apostille (nachfolgend: e-APP) des Justizministeriums Spaniens wurde mit dem Ziel entwickelt, eine Verwendung dieses Systems durch die Apostillen ausstellenden und zum Zuständigkeitsbereich dieses Ministeriums gehörenden Behörden sowie durch andere zur Apostillierung öffentlicher Urkunden aus anderen Bereichen befugte Einrichtungen zu ermöglichen. Zu diesem Zweck wurde ein Modulsystem unter Verwendung der Paradigmen „Dependency Injection" („Konnektoren") und „Serviceorientierung" entwickelt, welche es gestatten, auf einfache Art und Weise bestehende Komponenten durch andere auszutauschen, die den technologischen Bedürfnissen und Anforderungen jener Behörde entsprechen, in der dieses System eingeführt werden soll. Des Weiteren bietet es eine offene Architektur, die eine hochgradige Interoperabilität ermöglicht.

In diesem Dokument soll über die Möglichkeiten einer Einführung des von Justizministerium Spaniens entwickelten e-APP-Systems in anderen mit Apostillierungsbefugnissen ausgestatteten Behörden anderer Bereiche sowohl auf nationaler als auch auf internationaler Ebene informiert werden.

2. BESCHREIBUNG DES E-APP-SYSTEMS

2.1. Allgemeine Beschreibung des Systems

- Das e-APP-System stellt eine Applikation in J2EE-Webarchitektur dar, die auf zentralen Servern basiert und zu der die Endbenutzer über einen Browser Zugang erlangen können. Das System setzt sich aus zwei Untersystemen zusammen:
  - **Untersystem zur Bearbeitung elektronischer Apostillen**. Es handelt sich hierbei um ein Untersystem für den internen Gebrauch durch die zur Bearbeitung der Apostillen beauftragten Benutzer. Die Benutzer haben über einen einfachen Browser Zugang zu diesem Untersystem.
  - **Untersystem zur Voröffentlichung im Internet**. Dieses Untersystem gestattet die Durchführung folgender Vorgänge per Internet mittels Verwendung eines Browsers:
    - Herunterladen einer ausgestellten elektronischen Apostille (durch den Antragsteller mit der entsprechenden Authentifizierung)
    - Abfrage des elektronischen Apostillenverzeichnisses (bei ausländischen Behörden, ohne Authentifizierung)
- Das System basiert auf Services, die es mit den nötigen Grundfunktionen ausstatten, wie z.B. die Plattform für die elektronische Signatur oder der Service zur Erzeugung von Verifizierungscodes, und stellt gleichzeitig Services zur Verfügung, um über die so genannte „Integrationsschicht“ (SOA-Architektur) eine Integration mit anderen Applikationen zu ermöglichen.
- Bei der Implementierung im Justizministerium Spaniens sind folgende Produkte und Technologien verwendet worden:
  - Applikation in Java über WAS 6.1
  - DBMS: Oracle 10g
  - Elektronische Signatur-Plattform: ASF von TB Solutions
  - Format der Dokumente: Adobe PDF
  - Standards für die elektronische Signatur:
    - PAdES zur Unterzeichnung von e-Apostillen
    - XAdES, CAdES und PAdES sind als Formate zur Unterzeichnung öffentlicher Urkunden zugelassen
- Technische Anforderungen an die Client-Stations, von denen aus das e-APP-System verwendet werden soll:
  - **Untersystem zur Bearbeitung**: Dieses Untersystem ist vom Verwaltungspersonal der an Bearbeitungs- und Ausgabeprozessen von elektronischen Apostillen beteiligten Behörden zu verwenden. Die Softwareanforderungen an die Client-Stations sind hierfür:
    - Betriebssystem: Windows 2000, XP, Vista oder 7
    - Browser: Microsoft Internet Explorer 7 oder 8
    - PDF-Betrachterprogramm: Adobe Reader X
    - Software für Smartcard-Lesegeräte X509 v3

- **Untersystem zur Veröffentlichung im Internet.** Dieses Untersystem wird von den Antragstellern elektronischer Apostillen verwendet, die somit die Möglichkeit erhalten, die Apostillen online herunterzulagen. Des Weiteren kann es von ausländischen Empfängerbehörden dieser Apostillen benutzt werden, um eine Online-Prüfung letzterer vorzunehmen. In diesen Fällen bestehen folgende Software-Anforderungen für die Client-Stations:
  - Betriebssystem: Windows 2000, XP, Vista, 7 oder Linux
  - Browser: Microsoft Internet Explorer 7 oder 8, Mozilla Firefox 2.0 oder höher, Google Chrome 10
  - PDF-Betrachterprogramm: Adobe Reader X

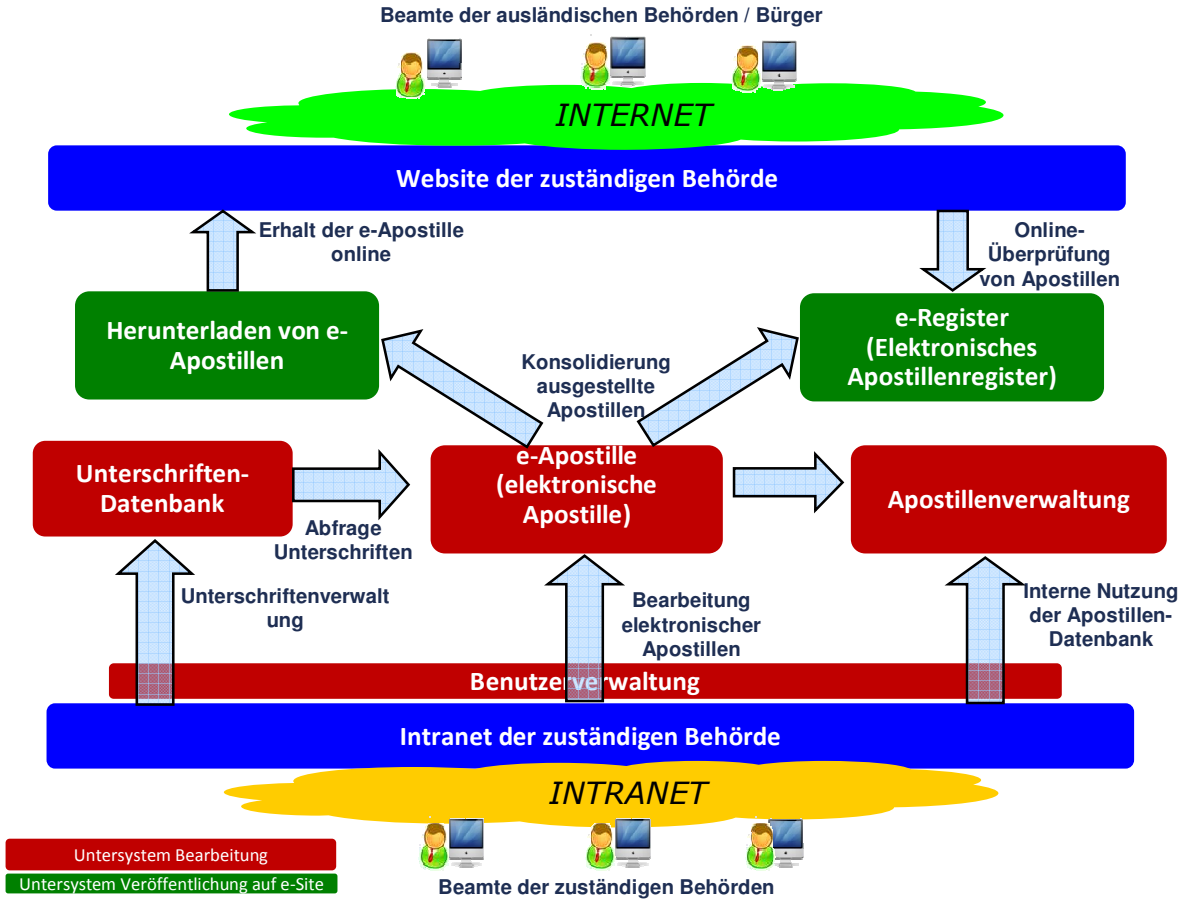
2.2. Funktionelle Zerlegung

Die beiden Untersysteme, aus denen sich das e-APP-System zusammensetzt (Untersystem zur Bearbeitung und Untersystem zur Veröffentlichung per Internet) lassen sich zu Funktionszwecken in verschiedene Module zerlegen:

- **Untersystem Bearbeitung**:
  - Modul Ausstellung elektronischer Apostillen (e-Apostille)
    - Erzeugung und elektronische Signatur der elektronischen Apostillen.
  - Modul Unterschriften-Datenbank
    - Anmeldung, Abmeldung und Änderung digitalisierter holografischer Unterschriften und/oder elektronischer Signaturen von Behörden, deren Unterschrift von der apostillierenden Behörde erkannt und überprüft werden soll.
  - Modul Apostillen-Verwaltung
    - Interne Abfrage und Aufbewahrung von Daten der Apostillen-Anträge
  - Modul Benutzerverwaltung
    - Anmeldung, Abmeldung, Änderung und Zuordnung von Benutzerprofilen der einzelnen Module des Untersystems Bearbeitung.

- **Untersystem Veröffentlichung im Internet**:
  - Modul elektronisches Apostillenregister (e-Register).
    - Elektronisches Register, in dem alle sowohl in Papierform als auch elektronisch ausgestellten Apostillen gespeichert und klassifiziert werden; dieses Register ist per Internet einsehbar
  - Modul Herunterladen von elektronischen Apostillen („elektronisches Erscheinen")
    - Gestattet dem Antragsteller das Herunterladen der elektronischen Apostille per Internet

Die erwähnten Module und deren Interaktion lassen sich in folgendem Diagramm darstellen:



Das e-APP-System verfügt des Weiteren über eine Integrationsschicht in Form von Webservices, die den Zugang zu den Funktionen des Systems von anderen Integrationssystemen aus gestattet. Innerhalb dieser Integrationsschicht lassen sich zwei Webservices unterscheiden:

- Webservice zur Integration mit e-Apostille (Herstellung von Bearbeitungsflüssen von Apostillen). Service zur Integration mit externen Applikationen zur Erzeugung elektronischer öffentlicher Urkunden

- Webservice zur Integration mit e-Register: Aufnahme von Apostillen ins e-Register. Ermöglicht eine von der e-Apostille unabhängige Nutzung des e-Registers.

2.3. Funktionsbeschreibung

In diesem Kapitel soll die grundlegende Funktionsweise des e-APP-Systems beschrieben werden.

Untersystem Bearbeitung von e-Apostillen

Modul e-Apostille:

Bearbeitung der Apostille auf einer in Papierform vorliegenden öffentlichen Urkunde

- Der Antragsteller legt die öffentliche Urkunde in Papierform vor
- Ein Benutzer in der Bearbeitung (mit Funktion zur „Aufnahme von Anträgen") erstellt einen neuen Antrag auf Apostillierung und trägt die diesbezüglichen Daten ein
- Das Dokument kann wahlweise mittels Scanner digitalisiert werden. In diesem Fall wird die digitalisierte öffentliche Urkunde als Anhang an die elektronische Apostille angefügt.
- Der Benutzer in der Bearbeitung registriert den Antrag. Der Antrag durchläuft daraufhin die nächste Phase, nämlich die „Validierung der Unterschrift der öffentlichen Urkunde". Von der Applikation wird ein Antragsnachweis erstellt, der vom Bearbeiter ausgedruckt und dem Antragsteller ausgehändigt wird.
- Ein Benutzer mit „Validierungsfunktion" (es kann sich um den gleichen Benutzer wie bei der Registrierung des Antrags handeln) validiert die holografische Unterschrift und/oder den Stempel der öffentlichen Urkunde. Zu diesem Zweck kann er von der Applikation aus die (digitalisierte) öffentliche Urkunde sowie die Abbildungen der Unterschrift und des Stempels des Unterzeichnenden einsehen. Ist alles korrekt, kann der Benutzer den Antrag als „gültig" markieren. Der Antrag befindet sich nunmehr in der Phase „validiert", d.h. für gültig erklärt.
- Ein Benutzer mit „Unterzeichnerfunktion" (es kann sich um den gleichen Benutzer wie bei der Validierung des Antrags handeln) stellt die Urkunde über die elektronische Apostille aus und unterzeichnet sie. Es handelt sich um ein Dokument, das als PDF-Datei im von der Konferenz von Den Haag für die Apostille vorgeschlagenen dreisprachigen Format (spanisch – englisch – französisch) vorliegt und, sofern es digitalisiert worden ist, als Anhang die Bilddatei der digitalisierten öffentlichen Urkunde enthält. Dieses wird mit einer PAdES-Unterschrift versehen. Ist die Unterzeichnung erfolgt, wird der Antrag auf Apostille in die Phase „ausgestellt" überführt.
- Ein Benutzer in der Bearbeitung (mit Funktion zur „Antragszustellung") greift auf die Apostille zu und stellt sie dem Antragsteller auf dem dafür ausgewählten Weg zu:
    - Durch Erscheinen: Der Bearbeiter muss den Antragsteller (per E-Mail oder telefonisch) benachrichtigen, dass die elektronische Apostille am Sitz der Behörde zur Abholung bereit liegt. Die Apostille kann entweder in elektronischem Format (auf Memory-Stick o.ä.) oder in Papierform ausgehändigt werden. Ist die Apostille vom Antragsteller abgeholt worden, markiertder Bearbeiter den Antrag in der Applikation als „zugestellt".
    - Per Post: Diese Möglichkeit besteht nur dann, wenn die Apostille zusammen mit der apostillierten öffentlichen Urkunde in Papierform ausgehängt wird. Ist der Versand per Post erfolgt, markiert der Bearbeiter den Antrag in der Applikation als „zugestellt".

- Durch „elektronisches Erscheinen” (sofern es vom Benutzer ausdrücklich so angefordert worden ist). In diesem Fall muss der Bearbeiter lediglich die Taste „Zustellen” betätigen, woraufhin die Applikation die Apostille sofort vom Modul „Herunterladen von Apostillen” des Untersystems Veröffentlichung im Internet aus zur Verfügung stellt.

Bearbeitung der Apostille auf elektronischer öffentlicher Urkunde

- Der Antragsteller legt die öffentliche Urkunde in elektronischem Format (Memory-Stick, o.ä.) vor.
- Ein Benutzer in der Bearbeitung (mit Funktion zur „Aufnahme von Anträgen") erstellt einen neuen Antrag auf Apostillierung und trägt die entsprechenden Daten ein. Die Datei des elektronischen öffentlichen Dokuments wird an diesen Antrag angehängt.
- Der Antrag wird vom Benutzer in der Bearbeitung registriert. Der Antrag durchläuft daraufhin die nächste Phase, nämlich die „Validierung der Unterschrift der öffentlichen Urkunde". Von der Applikation wird ein Antragsnachweis erstellt, der vom Bearbeiter ausgedruckt und dem Antragsteller ausgehändigt wird.
- Die elektronische Signatur der öffentlichen Urkunde wird vom System sofort automatisch validiert. Ist diese korrekt, wird in der Unterschriften-Datenbank nach dem Zertifikat der Unterzeichnung der öffentlichen Urkunde gesucht. Wird ein gültiger Unterzeichner für dieses Zertifikat gefunden, wird der Antrag für zulässig erklärt. Kann die Unterschrift jedoch nicht validiert oder das Zertifikat in der Unterschriften-Datenbank nicht aufgefunden werden, wird der Antrag „abgelehnt".
- Ein Benutzer mit „Unterzeichnerfunktion" stellt die Urkunde über die elektronische Apostille aus und unterzeichnet sie. Es handelt sich um ein Dokument, das als PDF-Datei im von der Konferenz von Den Haag für die Apostille vorgeschlagenen dreisprachigen Format (spanisch – englisch – französisch) vorliegt und als Anhang die Bilddatei der digitalisierten öffentlichen Urkunde enthält. Dieses wird mit einer PAdES-Unterschrift versehen. Ist die Unterzeichnung erfolgt, wird der Antrag auf Apostille in die Phase „ausgestellt" überführt.
- Ein Benutzer in der Bearbeitung (mit Funktion zur „Antragszustellung") greift auf die Apostille zu und stellt sie dem Antragsteller auf dem dafür ausgewählten Weg zu:
    - Durch Erscheinen: Der Bearbeiter muss den Antragsteller (per E-Mail oder telefonisch) benachrichtigen, dass die elektronische Apostille am Sitz der Behörde zur Abholung bereit liegt. Die Apostille kann lediglich in elektronischem Format (auf Memory-Stick o.ä.) ausgehändigt werden. Ist die Apostille vom Antragsteller abgeholt worden, markiert der Bearbeiter den Antrag in der Applikation als „zugestellt".
    - Durch „elektronisches Erscheinen" (sofern es von Benutzer ausdrücklich so angefordert wurde). In diesem Fall muss der Bearbeiter lediglich die Taste „Zustellen" betätigen, woraufhin die Applikation die Apostille sofort vom Modul „Herunterladen von Apostillen" des Untersystems Veröffentlichung im Internet aus zur Verfügung stellt.

Modul Unterschriftenverwaltung:

Dieses Modul ermöglicht die Wartung der Unterschriften-Datenbank, was folgende Vorgänge beinhaltet:

- Anmeldung einer neuen Unterschrift mit den Angaben bezüglich des Namens, der Funktion, der Behörde und des Gültigkeitszeitraums. Des Weiteren ist die Art der Unterschrift anzugeben (holografisch oder digital). Davon ausgehend sind der Eintragung ein bzw. zwei Dateien anzuhängen:
    - Im Fall der holografischen Unterschrift ist eine Datei mit der digitalisierten holografischen Unterschrift, bzw. eine Datei mit dem digitalisierten Stempel, oder beide anzuhängen. Diese Bilddateien müssen im JPEG-Format vorliegen.

  - Handelt es sich um eine digitale Unterschrift, ist eine Datei im DER-Format mit den elektronischen Kenndaten des Unterzeichnenden anzuhängen (Export im DER-Format des öffentlichen Teils der Urkunde, der zur Unterzeichnung verwendet wird)
- Suche und Abfrage von Unterschriften nach verschiedenen Suchkriterien
- Änderung der Unterschrift. Zuvor muss eine Abfrage zur Auswahl der zu ändernden Unterschrift erfolgt sein. Der nachfolgende Ablauf entspricht im Großen und Ganzen dem der Anmeldung.
- Löschung einer Unterschrift. Zuvor muss eine Abfrage zur Auswahl der zu löschenden Unterschrift erfolgt sein. Eine Unterschrift kann nur dann gelöscht werden, wenn sie zu diesem Zeitpunkt mit keinerlei Apostillenantrag in Verbindung steht.

Modul Apostillenverwaltung:

Dieses Modul ermöglicht die Verwaltung der Datenbank Apostillenanträge, einschließlich der Durchführung der folgenden Vorgänge:

- Suche und Abfrage von Anträgen nach verschiedenen Suchkriterien
- Änderung eines Antrags. Zuvor muss eine Abfrage zur Auswahl des abzuändernden Antrags erfolgt sein. Ein Antrag kann nur dann abgeändert werden, wenn er noch nicht die Phase „validiert" erreicht hat.
- Löschung eines Antrags. Zuvor muss eine Anfrage zur Auswahl des zu löschenden Antrags erfolgt sein. Ein Antrag kann nur dann gelöscht werden, wenn er noch nicht die Phase "ausgestellt" erreicht hat.
- Erstellung einer Auflistung der Anträge, wodurch das herkömmliche Apostillen-Register in Papierform ersetzt werden kann. Zuvor muss eine Abfrage zur Auswahl der in das Verzeichnis auszunehmenden Anträge erfolgt sein.

Modul Benutzerverwaltung:

Dieses Modul ermöglicht die Verwaltung der Systembenutzer, einschließlich der Durchführung der folgenden Vorgänge:

- Anmeldung eines Benutzers mit Angabe seiner personenbezogenen Daten, seines anfänglichen Passwords und der Befugnisse
- Suche und Abfrage von Benutzern nach verschiedenen Suchkriterien
- Änderung eines Benutzers. Zuvor muss eine Abfrage zur Auswahl des zu ändernden Benutzers erfolgt sein. Die weitere Vorgehensweise entspricht im Großen und Ganzen der der Anmeldung.
- Abmeldung eines Benutzers. Zuvor muss eine Abfrage zur Auswahl des abzumeldenden Benutzers erfolgt sein.

<u>Untersystem Veröffentlichung im Internet:</u>

Modul elektronisches Register der Apostillen (e-Register):

Dieses Modul gestattet es den ausländischen Empfängerbehörden der Apostillen (oder jeglichen anderen Interessenten), verschiedene Vorgänge zur Verifizierung der Apostillen durchzuführen. Um diese Aufgaben ausführen zu können, muss der Interessent drei Angaben zur Identifizierung der Apostille (welche in dieser enthalten sind) eingeben:

- Apostillennummer
- Ausstellungsdatum
- Verifizierungscode

Der „Verifizierungscode" ist ein alfanumerischer Code, durch den die Urkunde der elektronischen Apostille eindeutig identifiziert und letztere mit der unterzeichnenden öffentlichen Verwaltung in Verbindung gesetzt werden kann. Des Weiteren gestattet er, die Vollständigkeit und Authentizität der Apostille durch Zugang auf die Website dieser Behörde zu überprüfen. Es handelt sich um eine in der spanischen Gesetzgebung enthaltene Rechtsfigur, durch die den Druckkopien der elektronischen Apostille der rechtliche Wert einer elektronisch unterzeichneten Urkunde übertragen wird. Soll das System der elektronischen Apostille in Staaten verwendet werden, die nicht über eine derartige Rechtsfigur verfügen, sind die Apostillen lediglich mit einem alfanumerischen Code, der die einmalige Identifizierung einer zu einem bestimmten Daten ausgestellten Apostille sicherstellt (d.h. der Code darf sich für ein gleiches Ausstellungsdatum nicht wiederholen) zu versehen.

Nach Eingabe dieser Angaben kann der Interessent:

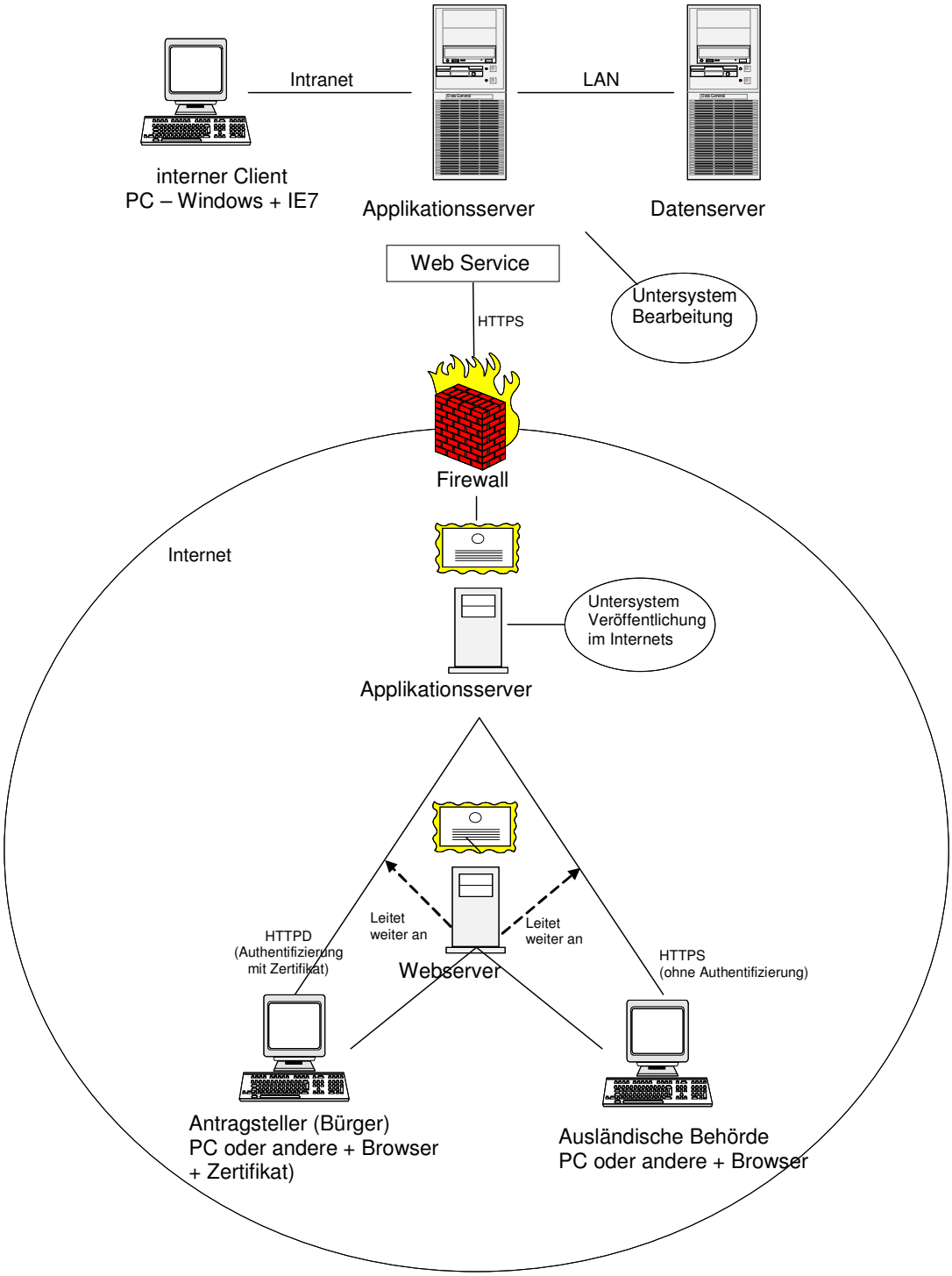
- **Die Apostille validieren**. Diese Option informiert den Interessenten lediglich darüber, dass im System tatsächlich eine Apostille mit den eingegebenen Angaben vorhanden ist, und gestattet ihm deren „Druckansicht" (Version ohne Unterschrift und ohne angehängte öffentliche Urkunde) zum visuellen Vergleich auf dem Bildschirm anzuzeigen.
- **Die öffentliche Urkunde validieren**. Hat der Interessent die öffentliche Urkunde sowie die Apostille in elektronischer Form erhalten, kann er über diese Option überprüfen, ob diese öffentliche Urkunde genau mit der apostillierten Urkunde übereinstimmt. Zu diesem Zweck muss der Interessent die Datei mit der öffentlichen Urkunde heraufladen. Das Modul e-Register erstellt daraufhin einen „elektronischen Fingerabdruck" (*Hash-Code*) für diese Datei und vergleicht diesen mit dem bereits gespeicherten und der Apostille zugeordneten Fingerabdruck. Stimmen beide überein, wird dem Interessenten mitgeteilt, dass es sich bei der hochgeladenen öffentlichen Urkunde genau um das apostillierte Dokument handelt.
- **Die Unterschrift der Apostille prüfen**. Verfügt der Empfänger einer elektronischen Apostille nicht über das entsprechende Tool zur Öffnung und Überprüfung der Unterschrift von mit PadES versehenen Dokumenten (z.B. Adobe Reader Version X) oder ist er aus sonstigen Gründen nicht in der Lage, mit seiner eigenen Software die Unterschrift der Apostille zu überprüfen, kann er die Datei mit der elektronischen Apostille „heraufladen". Über diese Option wird die Unterschrift der Apostille (einschließlich eines eventuellen Widerrufs des Zertifikats mit dem sie unterzeichnet wurde) im Server des Justizministeriums geprüft.

Modul Herunterladen von Apostillen:

Dieses Modul gestattet dem Antragsteller das Herunterladen der elektronischen Apostille per Internet nach deren Ausstellung. Zu diesem Zweck muss sich der Antragsteller mit einem Benutzernamen und einem Password identifizieren. Diese Angaben sind in der Antragsbescheinigung enthalten, die dem Antragsteller vom Bearbeiter ausgehändigt worden ist. Ist die Identifizierung erfolgt, kann der Antragsteller die Datei mit der elektronischen Apostille vom Browser aus herunterladen.

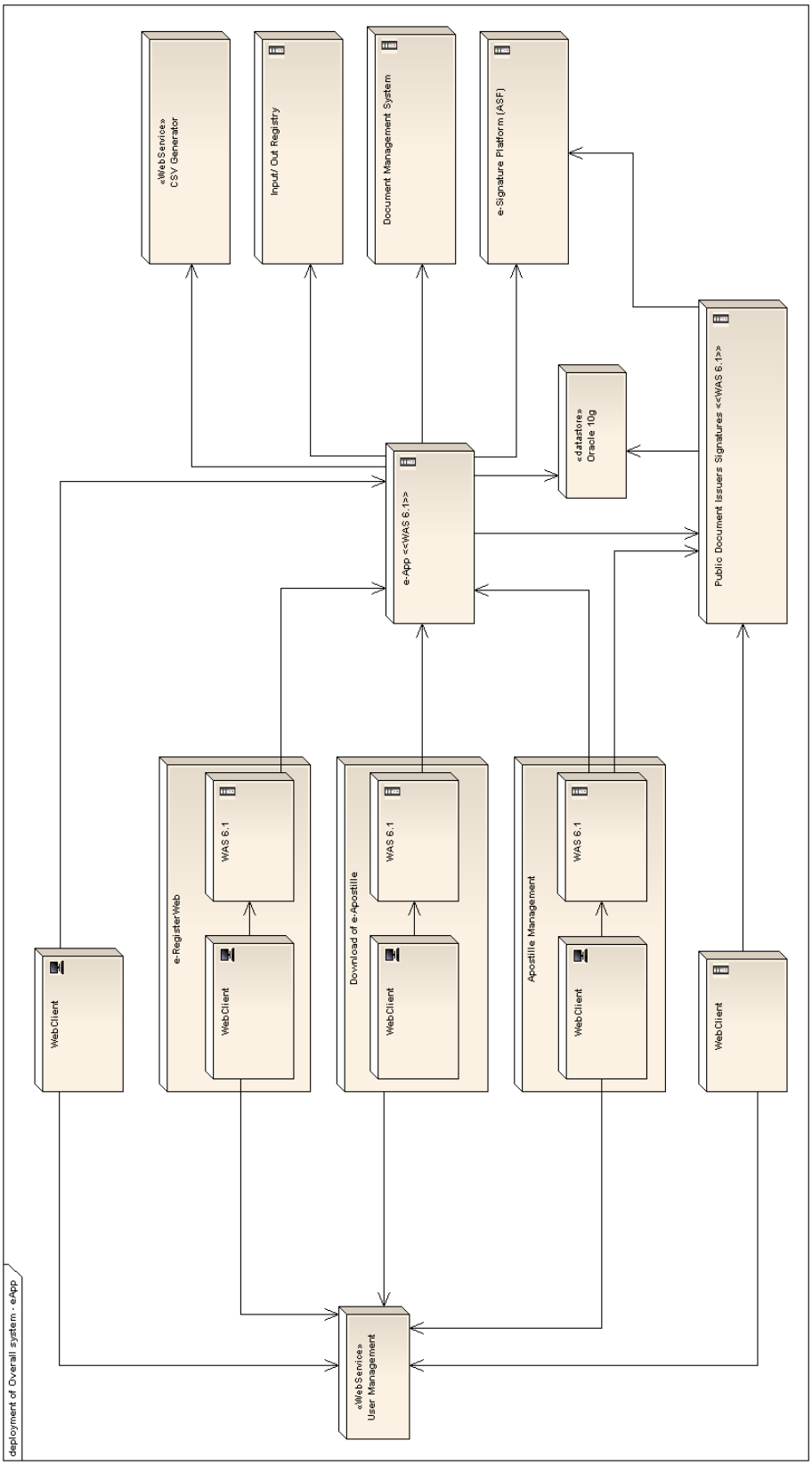
2.4. Architektur der Systemeinführung

Die Architekur der Einführung des e-APP-Systems lässt sich mit folgender Grafik darstellen:



2.4. Low-Level-Komponenten

Betrachtet man das System genauer, wird folgende Anordnung der Komponenten im e-APP-System deutlich.


«WebService»
CSV Generator
Input/ Out Registry
Document Management System
e-Signature Platform (ASF)
«datastore»
Oracle 10g
Public Document Issuers Signatures <<WAS 6.1>>
e-App <<WAS 6.1>>
WebClient
e-RegisterWeb
WAS 6.1
WebClient
Download of e-Apostille
WAS 6.1
WebClient
Apostille Management
WAS 6.1
WebClient
WebClient
«WebService»
User Management
deployment of Overall system - e-App

3. EXPORT DES E-APP-SYSTEMS

3.1. Austausch von Konnektoren

Dank seiner auf „Konnektoren" basierenden modularen Bauweise ermöglicht das System zur Bearbeitung elektronischer Apostillen den Austausch einiger bei seinem Aufbau verwendeten Module, welche die Kommunikation zwischen dem E-APP-System und anderen betrieblichen Systemen implementieren, durch andere, an die Bedürfnisse der Anwenderbehörde angepasste Module.
Ein „Konnektor" ist im Grunde genommen eine eine Schnittstelle herstellende Klasse, welche als eine Art „Vertrag" zwischen der Benutzerapplikation des Konnektors und dem Konnektor selbst agiert und somit eine standardisierte Kommunikation zwischen beiden herstellt. Somit ist es möglich, eine neue Implementierung des Konnektors zu schreiben, ohne dabei den Code der Applikation verändern zu müssen. Bedingung ist hierfür lediglich, dass der Konnektor die „vereinbarte" Schnittstelle beachtet.

Folgende Konnektoren sind austauschbar:

- **Konnektor mit Service zur Erzeugung von Verifizierungscodes**. Dieser Konnektor wird von der Anwendung e-Apostille in dem Moment geladen und benutzt, in dem ein Verifizierungscode zur Identifizierung der Apostille erzeugt werden muss. Die im Justizministerium Spaniens verwendete Implementierung greift auf einen horizontales Webservice des Ministeriums zurück, welcher den genannten Code erzeugt. Dieser Konnektor kann durch einen anderen, ebenfalls einen, wenn auch über ein anderes Verfahren, Verfizierungscode erzeugenden Konnektor ersetzt werden. Es besteht hierfür lediglich die Beschränkung, dass der Verifizierungscode eine alfanumerische Kette mit maximal 50 Zeichen sein muss, die die Eindeutigkeit der Dreiergruppe „Apostillennummer" / „Ausstellungsdatum" / „Verifizierungscode" sicherzustellen hat.
  Dieser Konnektor muss folgende Schnittstelle aufweisen:

```
public interface IConectorCSV extends IConector {
       public String obtenerCSV(String aplicacion, String tipoServicio)
                      throws GenericException;
}
```

  Das heißt, der Konnektor muss eine Zeichenklasse Java aufweisen, die eine Methode **obtenerCSV** implementiert, welche die beiden Argumente **aplicación** und **tipoServicio** erhält. Die Applikation Apostille benutzt diese Methode, indem für beide Argumente als Wert die Zeichenfolge „APOSTILLE" eingegeben wird. Diese Argumente werden vom im Justizministerium verwendeten Service verlangt, können jedoch bei jeglicher anderen Implementierung des Konnektors übergangen werden. Diese Methode sollte lediglich in der Lage sein, einen alfanumerischen Code mit maximal 50 Zeichen zu erzeugen, der für ein bestimmtes Datum und für eine Apostillennummer nur einmal existieren darf. So könnte z.B. ein Code erzeugt werden, der Datum-Uhrzeit des Systems (Jahr-Monat-Tag-Stunde-Minute-Sekunde-Millisekunde) + eine Zufallszahl enthält, wie im folgenden Code-Beispiel dargestellt wird:

```
public String obtenerCSV(String aplicacion, String tipoServicio) {
   String result=null;
   DateFormat dateFormat = new SimpleDateFormat("yyyyMMddHHmmssSSS");
   result = dateFormat.format(Calendar.getInstance().getTime()) +
Integer.toString((int)(Math.random()*9999));
   return result;
   }
}
```

- **Konnektor zum Dokumentenmanagementsystem.** Dieser Konnektor wird von der Applikation e-Apostille in dem Moment geladen und benutzt, wenn eine Speicherung von Urkunden im System und deren späterer Abruf erfolgen soll. Er wird sowohl zur vorübergehenden Speicherung von Urkunden (zur Apostillierung vorliegende elektronische bzw. digitalisierte öffentliche Urkunden, elektronische Apostille), als auch zur langfristigen Speicherung (Druckversion oder „Maske“ der Apostille) verwendet. Dieser Konnektor wird des Weiteren von der Applikation e-Register zum Abruf dieser Druckversion und deren Anzeige auf dem Bildschirm benutzt. Die derzeitige Implementierung speichert die Urkunden in einer BLOB-Spalte einer Oracle-Tabelle, obgleich das Modell eine Implementierung mit Speicherung der Urkunden in einer Ablage eines jeglichen Dokumentenmanagementsystems erlauben würde. Die einzige Bedingung ist jedoch, dass das Dokumentenmanagementsystem jeder Urkunde einen einmaligen Identifikator oder Lokalisator zuordnet, der in einer Zeichenfolge von 50 Zeichen gespeichert werden kann. Das Dokumentenmanagementsystem muss diesen Lokalisator im Zuge des Speicherungsvorgangs der Urkunde zurückgeben und nachfolgend deren Abruf durch die alleinige Verwendung dieses Lokalisators als Parameter ermöglichen.

  Dieser Konnektor muss folgende Schnittstelle aufweisen:

```
public interface IConectorGestorDocumental extends IConector {
       public String altaDocumento(InputStream inFile, String descripcion,
                      String extension) throws GenericException;
       public boolean eliminarDocumento(String uid) throws GenericException;
       public DatosDocumento leerDocumento(String uid) throws GenericException;
}
```

Das heißt, der Konnektor muss eine Zeichenklasse Java ausweisen, die die Methoden **altaDocumento**, **eliminarDocumento** und **leerDocumento** implementiert, welche folgende Argumente erhalten:

inFile: Datenstrom mit binärem Inhalt der zu speichernden Datei

descripcion: wörtliche Dateibeschreibung

extension: Erweiterung des Dateinamens (pdf, doc, txt, etc.)

uid: Einmaliger Identifikator der Datei in der Dokumentenablage

Die Methode **altaDocumento** muss den einmaligen Identifikator der Urkunde nach deren Speicherung in der Dokumentenablage zurückgeben.

Die Methode **eliminarDocumento** muss **true** zurückgeben, wenn es ihr gelungen ist, die Urkunde korrekt zu löschen oder anderenfalls eine Ausnahme einzurichten.

Die Methode **leerDocumento** muss ein Objekt der Klasse **DatosDocumento,** das sämtliche Informationen über die Urkunde, einschließlich deren binären Inhalt enthält, zurückgeben. Die Definition dieser Klasse ist in Anhang I dieses Dokuments einzusehen.

Das Justizministerium Spaniens kann interessierten Behörden den Konnektor zum Dokumentenmanagementsystem, welcher die Urkunden in BLOB-Fields der Datenbank speichert, zur Verfügung stellen. Das Justizministerium verwendet hierfür eine Oracle-Datenbank; dieser Konnektor kann jedoch auch mit anderen Datenbankmotoren, wie z.B. MySQL arbeiten. In diesem Fall wären am Konnektor die unter dem nachfolgenden Punkt 3.3. „Austausch des Datenbankmanagementsystems“ beschriebenen Anpassungen vorzunehmen.

- **Konnektor zur elektronischen Signaturplattform.** Dieser Konnektor wird von den Anwendungen e-Apostille und e-Register immer dann geladen und benutzt, wenn eine elektronische Signatur verifiziert werden muss. Im Fall der e-Apostille wird er zur Verifizierung der Unterschrift von elektronischen öffentlichen Urkunden und zur Verifizierung der Unterschrift der e-Apostille nach deren Ausstellung verwendet. Bezüglich des e-Registers wird er im Rahmen der Option „Verifizierung der Unterschrift von e-Apostillen” eingesetzt. Des Weiteren wird er vom Modul „Unterschriften” benutzt, um die Kenndaten (Aussteller und Seriennummer) der im DER-Format vorliegenden elektronischen Zertifikate, die zu den Unterzeichnern geladen werden, zu entnehmen. Dieser Konnektor kann somit durch einen anderen ersetzt werden, der eine andere Signaturserviceplattform verwendet, sofern letztere die folgenden Services zur Verfügung stellt:
  - Verifizierung der elektronischen Signatur einer Urkunde
  - Zugriff auf Daten (insbesondere Aussteller und Seriennummer) eines im DER-Format vorgelegten Zertifikats

  Es ist geplant, das System der e-Apostille in Zukunft mit der Möglichkeit der Unterzeichnung mit den so genannten „Behördenstempeln” auszustatten, d.h. einer Art der Unterschrift, die vom Server aus vorgenommen wird. Das heißt, nach Implementierung dieses Unterschriftverfahrens wird die Signaturserviceplattform auch zur Unterzeichnung von Apostillen verwendet werden. Derzeit wird der Konnektor für die Aufnahme dieser Funktion vorbereitet. Gegenwärtig muss dieser Konnektor folgende Schnittstelle aufweisen:

```
public interface IConectorPlataformaFirma extends IConector {
       public ValidacionResponse validarCertificado(InputStream certificadoDER,
                      String datosAdicionales) throws GenericException;

       public DatosValidarDocumentoFirmado[] validarDocumentoFirmado(
                      InputStream contenido, String datosAdicionales)
                      throws GenericException;
}
```

  Das heißt, der Konnektor muss eine Zeichenklasse Java aufweisen, die die Methoden **validarCertificado** und **validarDocumentoFirmado** implementieren. Diese Verfahren erhalten die folgenden Argumente:

  certificadoDER: Binärer Inhalt des Exports des öffentlichen Teils eines digitalen Zertifikats im DER-Format

  datosAdicionales: Datenstrom mit den möglichen Werten „XADES", „CADES" oder „PADES", durch den die Art der zu validierenden Urkunde angezeigt wird. Bei der Methode **validarCertificado** kommt dieses Argument eigentlich nicht zum Einsatz.

contenido: Binärer Inhalt der zu validierenden unterschriebenen Urkunde.

Durch die Methode **validarCertificado** wird eine Klasseninstanz **ValidacionResponse** mit Information über das als Argument gelieferte Zertifikat zurückgegeben. Das Verfahren **validarDocumentoFirmado** gibt ein Array von Klassenobjekten **DatosValidarDocumentoFirmado** mit Informationen zu der/den Unterschrift(en) der Urkunde zurück. Die Definition der beiden Klassen kann in Anhang I dieses Dokuments eingesehen werden.

3.2. Austausch des Unterschriftenmechanismus beim Client

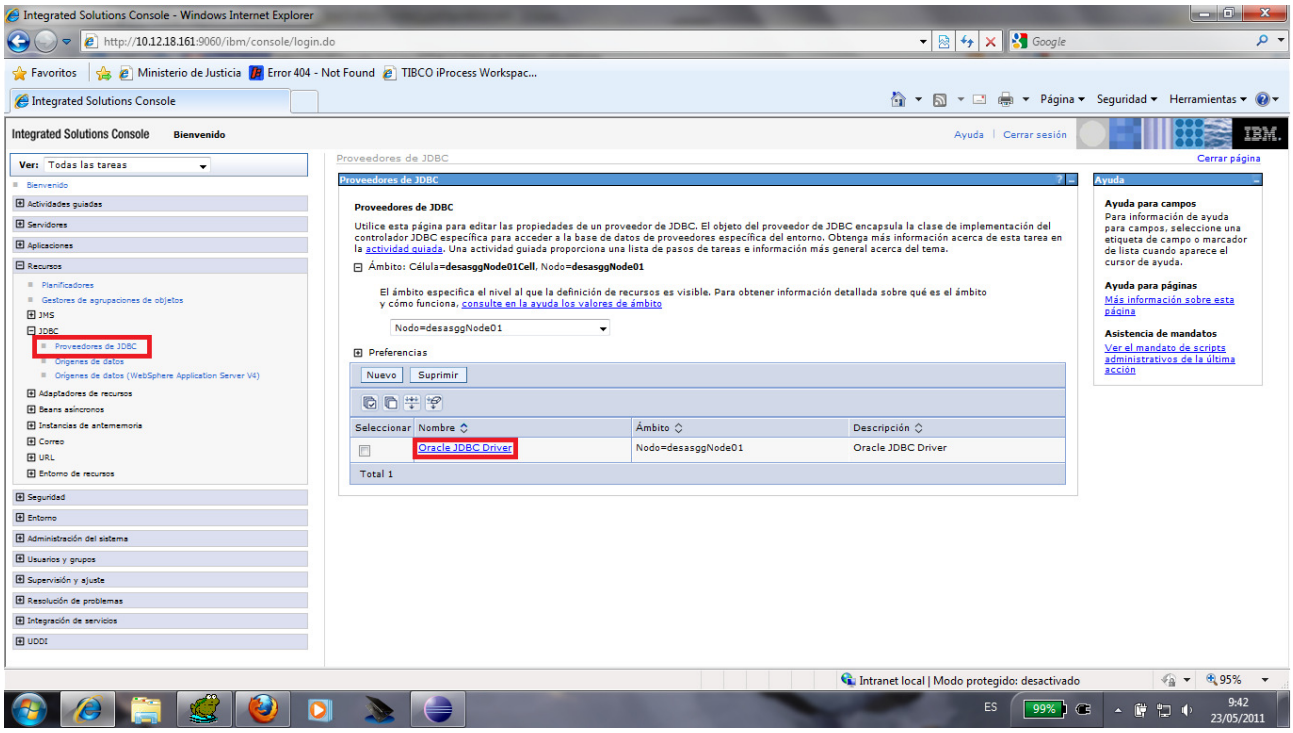
Die Applikation e-Apostille verwendet derzeit zur Unterzeichnung der Apostillen vom Client (Browser) aus ein Java-*Applet* mit dem Namen *WebSigner* von TB Solutions. Zur Interaktion mit diesem *Applet* wird eine *Javascript*-Schnittstelle verwendet. Es wäre denkbar, dieses Applet durch eine andere Unterschriftenlösung beim Client auszutauschen. Dafür wäre jedoch ein Austausch des Konnektors allein nicht ausreichend, sondern es müsste außerdem der *Javascript*-Code, der zum Zugang zum *Applet* verwendet worden ist, verändert werden. Sofern die gewählte Unterschriftenlösung ein Javascript-Programming-Interface aufweist, das dem von WebSigner ähnlich ist, kann diese Komponente ausgetauscht werden, obgleich dieser Austausch mit einem größeren Aufwand verbunden ist, da er eine Veränderung des Javascript-Codes der Webapplikation der e-Apostille erforderlich macht.
Wenn zukünftig die Möglichkeit besteht, die Apostillen über die Signaturserviceplattform im Server zu unterzeichnen, kann der Unterschriftenvorgang über den Konnektor zu dieser Plattform abgewickelt werden. In diesem Fall würde die Ersetzung des Konnektors zur Signaturplattform die Verwendung verschiedener Signaturplattformen zur digitalen Unterzeichnung von Apostillen im Server ermöglichen.

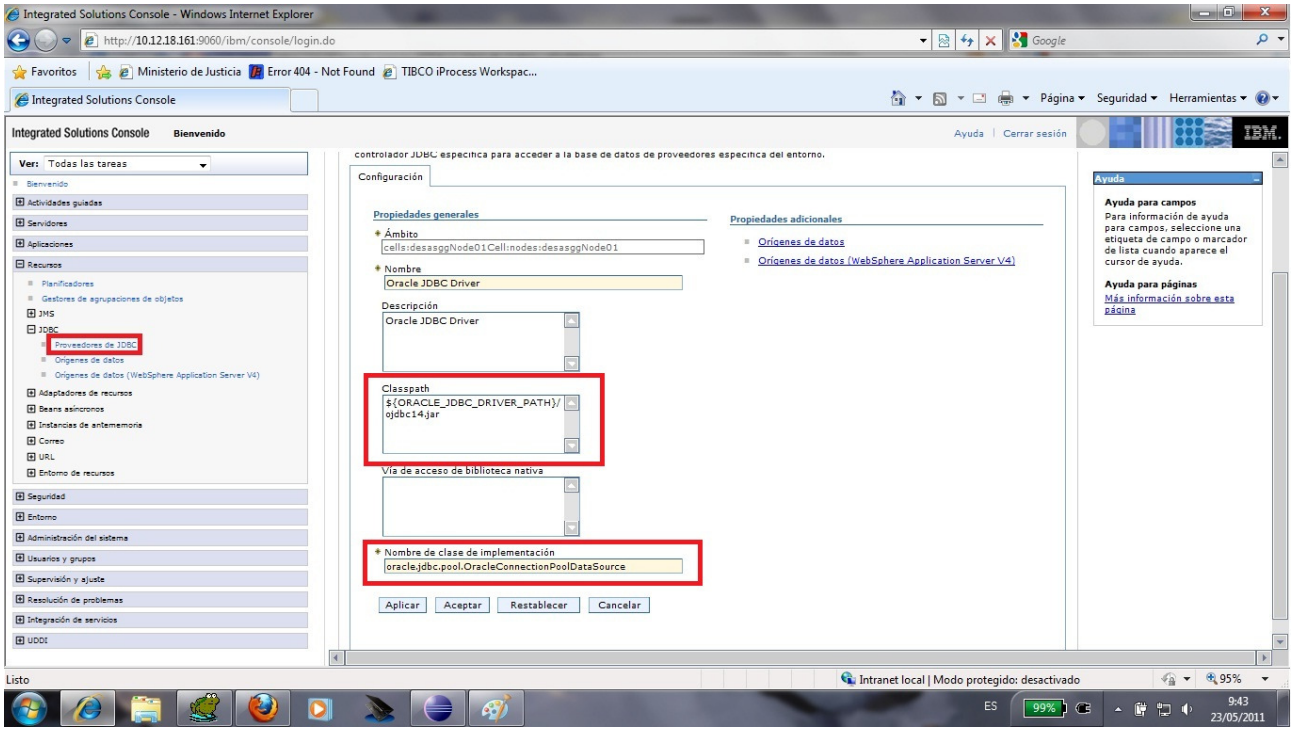
3.3. Austausch des Datenbankmanagementsystems

Alle Module des E-APP-Systems benutzen das Object-Relational Mapping von Hibernate. Diese Softwareschicht isoliert die Applikationen von den Spezifitäten des verwendeten Motors der relationalen Datenbank. Es könnte also jegliches anderes von Hibernate unterstütztes Datenbankmanagementsystem verwendet werden. In diesem Zusammenhang müssten lediglich folgende Änderungen vorgenommen werden:

- Konfiguration eines *JDBC-Drivers* und Errichtung einer *Datasource* mit Blick auf die zu verwendende Datenbank.
- Erstellung des *JNDI-Namens* der *Datasource* in den Dateien Eigenschaften
- Änderung des SQL-Dialekts des zu verwendenden Datenbankmanagementsystems in den Dateien Eigenschaften.

Die Konfigurierung des JDBC-Drivers erfolgt in der Konfigurierungsapplikation des WebSphere Application Server:





Im Fall von **MySQL** müssten z.B. unter *Classpath* der Pfad zum Jar-Modul, der die Bibliotheken von MySQL enthält: **`.../mysql.jar,`** sowie der *Name der Implementierungsklasse*: **`com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource`** eingegeben werden.

Daraufhin ist für diesen *JDBC-Driver* eine *Datasource* einzurichten und ihm ein *JNDI-Name* zuzuordnen.

Sowohl dieser JNDI-Name als auch der SQL-„Dialekt" müssen in den Dateien **applicationContext-configuration.xml** konfiguriert werden, die sich auf dem relativen Pfad: **services/src/main/resources/common/hibernate** innerhalb eines jeden, zum System der elektronischen Apostille gehörenden Projekt befinden. Für die Applikation eApostille müsste die Datei wie folgt geändert werden:

/eApp/eApp-services/src/main/resources/common/hibernate/applicationContext-configuration.xml

Die zu ändernden Eingaben sind gelb markiert:

```
<bean id="sessionFactory" class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
   <property name="dataSource" ref="dataSource" />
   <property name="mappingResources">
      <ref bean="mappingResources" />
   </property>
   <property name="hibernateProperties">
      <props>
         <prop key="hibernate.hbm2ddl.auto">none</prop>
         <prop key="hibernate.dialect">org.hibernate.dialect.Oracle10gDialect</prop>
         <prop key="hibernate.show_sql">false</prop>
      </props>
   </property>
</bean>
<bean id="dataSource" class="org.springframework.jndi.JndiObjectFactoryBean">
   <property name="resourceRef"><value>false</value></property>
   <property name="jndiName"><value>jdbc/eapp</value></property>
</bean>
```

Der *JNDI-Name* des Beispiels (jdbc/eapp) ist durch den der zum Einsatz kommenden *JDBL-Datasource* zugeordneten Namen zu ersetzen.

Bezüglich des Dialekts müsste z.B. bei der Verwendung von **MySQL 5.x** folgende Eingabe vorgenommen werden:

```
<prop key="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</prop>
```

Die Auflistung der unterstützten Dialekte ist der Dokumentation zu Hibernate einzusehen:

http://www.163jsp.com/help/hibernate32api/org/hibernate/dialect/Dialect.html

3.4. Integration über Webservices

Die Integrationsschicht des e-APP-Systems ermöglicht die Integration externer Applikationen mit vollständigen Systemmodulen. Konkret handelt es sich dabei um die Möglichkeit, eine Integration mit dem Modul e-Apostille und dem Modul e-Register vorzunehmen, wie im Folgenden beschrieben werden soll.

3.4.1. Integration mit e-Apostille

Der Webservice zur Integration mit e-Apostille publiziert Methoden, die es einer externen Applikation gestatten:

- Einen neuen Antrag auf e-Apostille zu erstellen und deren Bearbeitungsfluss einzuleiten (Methode `solicitarApostilla`)
- Den Bearbeitungsstatus einer e-Apostille zu erfahren und herauszufinden, ob sie bereits ausgestellt und unterzeichnet worden ist (Methode `consultarApostilla`)
- Eine bereits unterzeichnete e-Apostille aufzurufen (Methode `obtenerApostilla`)
- Einen (zuvor über den Webservice gestellten) Antrag auf eine Apostille zu annullieren, sofern diese noch nicht bearbeitet worden ist (Methode `anularApostilla`)

Über diesen Service kann eine elektronische öffentliche Urkunden erzeugende externe Applikation ihren Benutzern die Option einräumen, in ihrer Benutzerschnittstelle entsprechend zu „markieren", ob sie die beantragte öffentliche Urkunde gleich apostilliert haben möchten. In diesem Fall würde die Applikation nach Erstellung und Unterzeichnung der öffentlichen Urkunde einen Antrag auf elektronische Apostille für diese öffentliche Urkunde stellen, der über den normalen Weg der Applikation e-Apostille bearbeitet werden würde. Die externe Applikation könnte das e-APP-System regelmäßig fragen, ob ihr Antrag vollständig bearbeitet worden ist. Ist dies der Fall, kann sie die elektronische Apostille zurückholen und sie dem Endbenutzer vorlegen.
Die Definition dieses Services in WSDL ist in Anhang II dieses Dokuments enthalten.

3.4.2. Integration mit e-Register

Der Webservice zur Integration mit e-Apostille publiziert eine Methode, die es einer elektronische Apostillen erstellenden Applikation gestattet, diese in ein e-Register aufzunehmen. Zu diesem Zweck muss die externe Applikation mindestens die folgenden Elemente bereitstellen:

- Die Identifikationsdaten der elektronischen Apostille: Nummer, Ausstellungsdatum und Verifizierungscode
- Weitere im Übereinkommen von Den Haag zur Registrierung von Apostillen geforderte Mindestangaben:
    - Name des Unterzeichners der öffentlichen Urkunde
    - Dessen Funktion
    - Die den Stempel oder die Stempelmarke aufsetzende Behörde
- Eine PDF-Datei mit der „Druckversion" der elektronischen Apostille.

Wird die Funktion „hash der öffentlichen Urkunde verifizieren" im e-Register gewünscht, müssen neben der Integrationsmethode folgende Angaben geliefert werden:

- Eine nach einem der unterstützten Algorithmen erzeugte Hashfunktion der öffentlichen Urkunde
- Der Identifikationsname des verwendeten Algorithmus

Die unterstützten *Hash*-Algorithmen sind:

- SHA-1
- MD5

Damit die e-Register-Option „Unterschrift der e-Apostille verifizieren" verwendet werden kann, bedarf es einer Serviceplattform mit ihrem entsprechenden Konnektor.

Zum Zeitpunkt der Erstellung dieses Formulars lag die Definition dieses Services in WSDL noch nicht vor. Sobald sie verfügbar ist, kann sie den Interessenten übermittelt werden.

3.5. Anpassung von Ausgangstabellen

Neben den entsprechenden Software-Anpassungen bedarf es zum Export des e-APP-Systems des Justizministeriums Spaniens in andere Behörden einer Anpassung des Inhalts einiger Ausgangsdatentabellen mit Informationen bezüglich der Systemkonfiguration. Es handelt sich dabei um folgende Tabellen:

- Tabelle der Amtssitze (TC_SEDE). Hier sind die Daten der einzelnen Amtssitze der apostillierenden Behörden einzugeben, wofür die die Lösung einführende Behörde zuständig ist. Ihre Definition ist in Anhang III dieses Dokuments einzusehen.

3.6. Lokalisierung

Soll das System in einem nicht spanischsprachigen Land eingeführt werden, müssen sowohl die statischen Texte der Formulare als auch die Inhalte der Ausgangstabellen in die Sprache des Landes übersetzt werden, in dem das System eingeführt werden soll (Lokalisierungsprozess).

Zur Lokalisierung der statischen Texte sind die Dateien Eigenschaften (Textdateien), in denen sich diese Texte befinden und welche sich zusammen mit der Applikation im WAS-Server auffächern, abzuändern. Diese Dateien befinden sich auf dem relativen Pfad **`src/main/resources/locale`** eines jeden zum e-APP-System gehördenden Moduls. Für das Modul e-Apostille würde man z.B. die Datei Eigenschaften für eine jede Sprache auf dem Pfad: **`eApp/eApp-webapp/src/main/resources/locale`** `finden`.

Folgende Ausgangstabellen müssten dann angepasst werden:

- Tabelle der Länder (TC_PAIS)
- Tabelle des Bearbeitungsstatus (TC_ESTADO)

HINWEIS: Die Tabellen „Provinzen" (TC_PROVINCIA) und „Gemeinden" (TC_MUNICIPIO) kommen nur dann zur Anwendung, wenn „Spanien" als Land des Wohnsitzes gewählt wird. Bei einer Einführung in anderen Ländern müssen diese Tabellen also nicht angepasst bzw. mit territorialen Angaben des Landes versehen werden, in dem diese Lösung eingeführt werden soll. Der Ort und/oder die Provinz o.ä. des Wohnsitzes können in diesem Fall manuell eingegeben werden.
Ist die territoriale Gliederung des Landes, in dem die Lösung eingeführt werden soll, mit der Spaniens vergleichbar (Unterteilung in Provinzen und Gemeinden, oder vergleichbare Einheiten), kann im Applikationscode eine kleine Anpassung vorgenommen werden, damit diese Felder bei der Auswahl des entsprechenden Landes mit erscheinen. In diesem Fall müssten die Tabellen der Provinzen und Gemeinden mit den Informationen des jeweiligen Landes gespeist werden.
Die Definition dieser Tabellen ist in Anhang III dieses Dokuments einzusehen.

ANHANG I. DEFINITION VON KLASSEN

Klasse DatenDokument

```
public class DatosDocumento implements java.io.Serializable {
	private static final long serialVersionUID = -604263192269590408L;
	private String doUidDocumento;
	private String doDescripcion;
	private OutputStream doArchivo;
	private String doExtension;

	public DatosDocumento() {
	}

	public String getDoUidDocumento() {
		return this.doUidDocumento;
	}

	public void setDoUidDocumento(String doUidDocumento) {
		this.doUidDocumento = doUidDocumento;
	}

	public String getDoDescripcion() {
		return this.doDescripcion;
	}

	public void setDoDescripcion(String doDescripcion) {
		this.doDescripcion = doDescripcion;
	}

	public String getDoExtension() {
		return this.doExtension;
	}

	public void setDoExtension(String doExtension) {
		this.doExtension = doExtension;
	}

	public OutputStream getDoArchivo() {
		return doArchivo;
	}

	public void setDoArchivo(OutputStream doArchivo) {
		this.doArchivo = doArchivo;
	}
}
```

Klasse ValidierungAntwort

```
public class ValidacionResponse implements IDescribeError {
	private static final long serialVersionUID = 3567193432574376046L;
	private String codError;
	private String descError;
	private String issuer;
	private String subject;
	private String serialNumber;
	private Calendar validFrom;
	private Calendar validUntil;
	private int certState;

	public String getIssuer() {
		return issuer;
	}

	public void setIssuer(String issuer) {
		this.issuer = issuer;
	}

	public String getSerialNumber() {
		return serialNumber;
	}

	public void setSerialNumber(String serialNumber) {
		this.serialNumber = serialNumber;
	}

	public Calendar getValidFrom() {
		return validFrom;
	}

	public void setValidFrom(Calendar validFrom) {
		this.validFrom = validFrom;
	}

	public Calendar getValidUntil() {
		return validUntil;
	}

	public void setValidUntil(Calendar validUntil) {
		this.validUntil = validUntil;
	}

	public int getCertState() {
		return certState;
	}

	public void setCertState(int certState) {
		this.certState = certState;
	}


	public String getCodError() {
```

```
		return this.codError;
	}

	public String getDescError() {
		return this.descError;
	}

	public void setCodError(String codigo) {
		this.codError = codigo;
	}

	public void setDescError(String descripcion) {
		this.descError = descripcion;
	}

	public void setSubject(String subject) {
		this.subject = subject;
	}

	public String getSubject() {
		return subject;
	}
}
```

Klasse DatenValidierungUnterzeichnetesDokument

```
public class DatosValidarDocumentoFirmado implements IDescribeError {
	private static final long serialVersionUID = 3567193432574376046L;
	private String codError;
	private String descError;
	private String issuer;
	private String subject;
	private Calendar validFrom;
	private Calendar validUntil;
	private Calendar signDate;
	private String serialNumber;
	private int resultCode;

	public String getIssuer() {
		return issuer;
	}

	public void setIssuer(String issuer) {
		this.issuer = issuer;
	}

	public Calendar getValidFrom() {
		return validFrom;
	}

	public void setValidFrom(Calendar validFrom) {
		this.validFrom = validFrom;
	}

	public Calendar getValidUntil() {
		return validUntil;
	}

	public void setValidUntil(Calendar validUntil) {
		this.validUntil = validUntil;
	}

	public String getCodError() {
		return this.codError;
	}

	public String getDescError() {
		return this.descError;
	}

	public void setCodError(String codigo) {
		this.codError = codigo;
	}

	public void setDescError(String descripcion) {
		this.descError = descripcion;
	}

	public void setSubject(String subject) {
```

```
		this.subject = subject;
	}

	public String getSubject() {
		return subject;
	}

	public void setSignDate(Calendar signDate) {
		this.signDate = signDate;
	}

	public Calendar getSignDate() {
		return signDate;
	}

	public void setSerialNumber(String serialNumber) {
		this.serialNumber = serialNumber;
	}

	public String getSerialNumber() {
		return serialNumber;
	}

	public void setResultCode(int resultCode) {
		this.resultCode = resultCode;
	}

	public int getResultCode() {
		return resultCode;
	}
}
```

ANHANG II. DEFINITIONEN DER WEBSERVICES IN WSDL

WSDL des Services zur Integration mit e-Apostille

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://impl.ws.eApostille.eApp.apostilla.majus.ntj.mju"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://impl.ws.eApostille.eApp.apostilla.majus.ntj.mju"
xmlns:intf="http://impl.ws.eApostille.eApp.apostilla.majus.ntj.mju"
xmlns:tns1="http://utils.ws.eApostille.eApp.apostilla.majus.ntj.mju"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)-->
 <wsdl:types>
  <schema elementFormDefault="qualified"
targetNamespace="http://utils.ws.eApostille.eApp.apostilla.majus.ntj.mju"
xmlns="http://www.w3.org/2001/XMLSchema">
   <complexType name="InfConsulApostilla">
    <sequence>
     <element name="csv" nillable="true" type="xsd:string"/>
     <element name="numApostilla" nillable="true" type="xsd:string"/>
    </sequence>
   </complexType>
   <complexType name="DatosConsulApostilla">
    <sequence>
     <element name="autoridadApostillante" nillable="true" type="xsd:string"/>
     <element name="autoridadFirmante" nillable="true" type="xsd:string"/>
     <element name="calidadFirmante" nillable="true" type="xsd:string"/>
     <element name="codError" nillable="true" type="xsd:string"/>
     <element name="comparecenciaElectronica" type="xsd:boolean"/>
     <element name="cp" nillable="true" type="xsd:string"/>
     <element name="csv" nillable="true" type="xsd:string"/>
     <element name="descError" nillable="true" type="xsd:string"/>
     <element name="direccionSolicitante" nillable="true" type="xsd:string"/>
     <element name="emailSolicitante" nillable="true" type="xsd:string"/>
     <element name="estado" nillable="true" type="xsd:string"/>
     <element name="fechaAnulacion" nillable="true" type="xsd:dateTime"/>
     <element name="fechaCreacion" nillable="true" type="xsd:dateTime"/>
     <element name="fechaDescarga" nillable="true" type="xsd:dateTime"/>
     <element name="fechaEmision" nillable="true" type="xsd:dateTime"/>
     <element name="fechaFirma" nillable="true" type="xsd:dateTime"/>
     <element name="fechaNotificacion" nillable="true" type="xsd:dateTime"/>
     <element name="fechaValidacion" nillable="true" type="xsd:dateTime"/>
     <element name="fechaVigencia" nillable="true" type="xsd:dateTime"/>
     <element name="identificadorSolicitante" nillable="true" type="xsd:string"/>
     <element name="localizador" nillable="true" type="xsd:string"/>
     <element name="movRechazo" nillable="true" type="xsd:string"/>
     <element name="municipioSolicitante" nillable="true" type="xsd:string"/>
     <element name="nombreSolicitante" nillable="true" type="xsd:string"/>
     <element name="numApostilla" nillable="true" type="xsd:string"/>
     <element name="numRegEntrada" nillable="true" type="xsd:string"/>
     <element name="numRegSalida" nillable="true" type="xsd:string"/>
     <element name="organismo" nillable="true" type="xsd:string"/>
     <element name="paisDestino" type="xsd:int"/>
```

```
  <element name="paisDestinoDesc" nillable="true" type="xsd:string"/>
  <element name="paisSolicitante" type="xsd:int"/>
  <element name="paisSolicitanteDesc" nillable="true" type="xsd:string"/>
  <element name="provinciaSolicitante" type="xsd:int"/>
  <element name="provinciaSolicitanteDesc" nillable="true" type="xsd:string"/>
  <element name="telefonoSolicitante" nillable="true" type="xsd:string"/>
  <element name="tipoDocumento" type="xsd:boolean"/>
  <element name="url" nillable="true" type="xsd:string"/>
 </sequence>
</complexType>
<complexType name="InfSolApostilla">
 <sequence>
  <element name="DNI_solicitante" nillable="true" type="xsd:string"/>
  <element name="apAutoridadApostillante" nillable="true" type="xsd:string"/>
  <element name="apAutoridadFirmante" nillable="true" type="xsd:string"/>
  <element name="apCalidadFirmante" nillable="true" type="xsd:string"/>
  <element name="apComparecenciaElect" type="xsd:boolean"/>
  <element name="apCp" nillable="true" type="xsd:string"/>
  <element name="apDireccion" nillable="true" type="xsd:string"/>
  <element name="apEmail" nillable="true" type="xsd:string"/>
  <element name="apLocalizadorDpElect" nillable="true" type="xsd:string"/>
  <element name="apMunicipio" nillable="true" type="xsd:string"/>
  <element name="apOrganismo" nillable="true" type="xsd:string"/>
  <element name="apTelefono" nillable="true" type="xsd:string"/>
  <element name="apTipoFormatoDp" type="xsd:int"/>
  <element name="apUrlDpElect" nillable="true" type="xsd:string"/>
  <element name="apellidos_solicitante" nillable="true" type="xsd:string"/>
  <element name="descripcionDP" nillable="true" type="xsd:string"/>
  <element name="docPublico" nillable="true" type="xsd:base64Binary"/>
  <element name="extensionDP" nillable="true" type="xsd:string"/>
  <element name="idPais" type="xsd:int"/>
  <element name="idPaisDest" type="xsd:int"/>
  <element name="idProv" type="xsd:int"/>
  <element name="idSede" type="xsd:long"/>
  <element name="nombre_solicitante" nillable="true" type="xsd:string"/>
  <element name="usr_solicitante" nillable="true" type="xsd:string"/>
 </sequence>
</complexType>
<complexType name="DatosSolApostilla">
 <sequence>
  <element name="DNI_solicitante" nillable="true" type="xsd:string"/>
  <element name="apAutoridadApostillante" nillable="true" type="xsd:string"/>
  <element name="apAutoridadFirmante" nillable="true" type="xsd:string"/>
  <element name="apCalidadFirmante" nillable="true" type="xsd:string"/>
  <element name="apComparecenciaElect" type="xsd:boolean"/>
  <element name="apCp" nillable="true" type="xsd:string"/>
  <element name="apCsv" nillable="true" type="xsd:string"/>
  <element name="apDescripcionDp" nillable="true" type="xsd:string"/>
  <element name="apDireccion" nillable="true" type="xsd:string"/>
  <element name="apEmail" nillable="true" type="xsd:string"/>
  <element name="apHash" nillable="true" type="xsd:string"/>
  <element name="apLocalizadorDpElect" nillable="true" type="xsd:string"/>
  <element name="apMotivoRechazo" nillable="true" type="xsd:string"/>
  <element name="apMunicipio" type="xsd:long"/>
  <element name="apNumRegEntrada" nillable="true" type="xsd:string"/>
  <element name="apNumRegSalida" nillable="true" type="xsd:string"/>
```

```
      <element name="apNumeroApostilla" nillable="true" type="xsd:string"/>
      <element name="apOrganismo" nillable="true" type="xsd:string"/>
      <element name="apTelefono" nillable="true" type="xsd:string"/>
      <element name="apTipoFormatoDp" type="xsd:int"/>
      <element name="apUrlDpElect" nillable="true" type="xsd:string"/>
      <element name="apellidos_solicitante" nillable="true" type="xsd:string"/>
      <element name="codError" nillable="true" type="xsd:string"/>
      <element name="descError" nillable="true" type="xsd:string"/>
      <element name="descPais" nillable="true" type="xsd:string"/>
      <element name="descPaisDest" nillable="true" type="xsd:string"/>
      <element name="descProv" nillable="true" type="xsd:string"/>
      <element name="docPublico" nillable="true" type="xsd:base64Binary"/>
      <element name="extensionDP" nillable="true" type="xsd:string"/>
      <element name="idPais" type="xsd:int"/>
      <element name="idPaisDest" nillable="true" type="xsd:string"/>
      <element name="idProv" type="xsd:int"/>
      <element name="idSede" type="xsd:int"/>
      <element name="nombre_solicitante" nillable="true" type="xsd:string"/>
     </sequence>
    </complexType>
    <complexType name="InfAnulApostilla">
     <sequence>
      <element name="csv" nillable="true" type="xsd:string"/>
      <element name="numApostilla" nillable="true" type="xsd:string"/>
     </sequence>
    </complexType>
    <complexType name="DatosAnulApostilla">
     <sequence>
      <element name="codError" nillable="true" type="xsd:string"/>
      <element name="descError" nillable="true" type="xsd:string"/>
      <element name="resultadoAnulacion" type="xsd:boolean"/>
     </sequence>
    </complexType>
    <complexType name="InfObtenerApostilla">
     <sequence>
      <element name="csv" nillable="true" type="xsd:string"/>
      <element name="numApostilla" nillable="true" type="xsd:string"/>
     </sequence>
    </complexType>
    <complexType name="DatosObtenerApostilla">
     <sequence>
      <element name="codError" nillable="true" type="xsd:string"/>
      <element name="descError" nillable="true" type="xsd:string"/>
      <element name="fichero" nillable="true" type="xsd:string"/>
     </sequence>
    </complexType>
   </schema>
   <schema elementFormDefault="qualified"
 targetNamespace="http://impl.ws.eApostille.eApp.apostilla.majus.ntj.mju"
 xmlns="http://www.w3.org/2001/XMLSchema">
    <import namespace="http://utils.ws.eApostille.eApp.apostilla.majus.ntj.mju"/>
    <element name="consulta" type="tns1:InfConsulApostilla"/>
    <element name="consultarApostillaReturn" type="tns1:DatosConsulApostilla"/>
    <element name="solicitud" type="tns1:InfSolApostilla"/>
    <element name="solicitarApostillaReturn" type="tns1:DatosSolApostilla"/>
    <element name="anulacion" type="tns1:InfAnulApostilla"/>
```

```
    <element name="anularApostillaReturn" type="tns1:DatosAnulApostilla"/>
    <element name="apostilla" type="tns1:InfObtenerApostilla"/>
    <element name="obtenerApostillaReturn" type="tns1:DatosObtenerApostilla"/>
   </schema>
  </wsdl:types>

    <wsdl:message name="anularApostillaResponse">
       <wsdl:part element="impl:anularApostillaReturn" name="anularApostillaReturn">
       </wsdl:part>
    </wsdl:message>

    <wsdl:message name="anularApostillaRequest">
       <wsdl:part element="impl:anulacion" name="anulacion">
       </wsdl:part>
    </wsdl:message>

    <wsdl:message name="solicitarApostillaRequest">
       <wsdl:part element="impl:solicitud" name="solicitud">
       </wsdl:part>
    </wsdl:message>

    <wsdl:message name="consultarApostillaRequest">
       <wsdl:part element="impl:consulta" name="consulta">
       </wsdl:part>
    </wsdl:message>

    <wsdl:message name="solicitarApostillaResponse">
       <wsdl:part element="impl:solicitarApostillaReturn" name="solicitarApostillaReturn">
       </wsdl:part>
    </wsdl:message>

    <wsdl:message name="consultarApostillaResponse">
       <wsdl:part element="impl:consultarApostillaReturn" name="consultarApostillaReturn">
       </wsdl:part>
    </wsdl:message>

    <wsdl:message name="obtenerApostillaRequest">
       <wsdl:part element="impl:apostilla" name="apostilla">
       </wsdl:part>
    </wsdl:message>

    <wsdl:message name="obtenerApostillaResponse">
       <wsdl:part element="impl:obtenerApostillaReturn" name="obtenerApostillaReturn">
       </wsdl:part>
    </wsdl:message>
```

```
   <wsdl:portType name="ServicioSolicitarApostillas">
      <wsdl:operation name="consultarApostilla" parameterOrder="consulta">
         <wsdl:input message="impl:consultarApostillaRequest"
name="consultarApostillaRequest">
       </wsdl:input>

         <wsdl:output message="impl:consultarApostillaResponse"
name="consultarApostillaResponse">
       </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="solicitarApostilla" parameterOrder="solicitud">

         <wsdl:input message="impl:solicitarApostillaRequest"
name="solicitarApostillaRequest">
       </wsdl:input>
         <wsdl:output message="impl:solicitarApostillaResponse"
name="solicitarApostillaResponse">
       </wsdl:output>
      </wsdl:operation>

      <wsdl:operation name="anularApostilla" parameterOrder="anulacion">
         <wsdl:input message="impl:anularApostillaRequest" name="anularApostillaRequest">
       </wsdl:input>
         <wsdl:output message="impl:anularApostillaResponse" name="anularApostillaResponse">
       </wsdl:output>
      </wsdl:operation>

      <wsdl:operation name="obtenerApostilla" parameterOrder="apostilla">
         <wsdl:input message="impl:obtenerApostillaRequest" name="obtenerApostillaRequest">
       </wsdl:input>
         <wsdl:output message="impl:obtenerApostillaResponse"
name="obtenerApostillaResponse">
       </wsdl:output>
      </wsdl:operation>

   </wsdl:portType>

   <wsdl:binding name="ServicioSolicitarApostillasSoapBinding"
type="impl:ServicioSolicitarApostillas">
      <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="consultarApostilla">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="consultarApostillaRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="consultarApostillaResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
```

```
      </wsdl:operation>
      <wsdl:operation name="solicitarApostilla">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="solicitarApostillaRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="solicitarApostillaResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>

      <wsdl:operation name="anularApostilla">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="anularApostillaRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="anularApostillaResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>

      <wsdl:operation name="obtenerApostilla">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="obtenerApostillaRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="obtenerApostillaResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>

   </wsdl:binding>

   <wsdl:service name="ServicioSolicitarApostillasService">
      <wsdl:port binding="impl:ServicioSolicitarApostillasSoapBinding"
name="ServicioSolicitarApostillas">
         <wsdlsoap:address
location="http://localhost:8080/ServicioSolicitarApostillasWS/services/ServicioSolicitarApost
illas"/>
      </wsdl:port>
   </wsdl:service>

</wsdl:definitions>
```

ANHANG III. DEFINITIONEN VON RELATIONALEN TABELLEN

Tabelle der Amtssitze

TC_SEDE				
Es werden die Daten der „Amtssitze" der apostillierenden Behörden gespeichert.				
Spalte	**Datenart**	**Beschreibung**	**Obligatorisch**	**Werte**
SE_CLAVE	NUMBER (12)	Einmaliger Identifikator eines Amtssitzes	JA	
SE_NOMBRE	VARCHAR2 (50)	Code des Amtssitzes (aus der vom Justizministerium zur Verfügung gestellten Seite des Kompatibilitätstests)	JA	
SE_DESCRIPCION	VARCHAR2 (150)	Beschreibung des Amtssitzes	NEIN	
SE_MUNICIPIO	VARCHAR2 (100)	Gemeinde des Amtssitzes	JA	
SE_TELEFONO	VARCHAR2 (35)	Telefonnummer des Amtssitzes	NEIN	
SE_FAX	VARCHAR2 (35)	Faxnummer des Amtssitzes	NEIN	
SE_DIRECCION	VARCHAR2 (250)	Postanschrift des Amtssitzes	NEIN	
SE_CP	VARCHAR2 (5)	Postleitzahl des Amtssitzes	NEIN	
SE_EMAIL	VARCHAR2 (150)	E-Mail-Adresse des Amtssitzes	NEIN	

Tabelle des Bearbeitungsstatus

TC_ESTADO				
Hier werden die verschiedenen Bearbeitungsstatus eines Apostillenantrags /einer Apostille gespeichert				
Spalte	**Datenart**	**Beschreibung**	**Obligatorisch**	**Werte**
ES_CLAVE	NUMMER (2)	Einmaliger Identifikator eines Bearbeitszustands einer Apostille	JA	Werte: - 1: BEGONNEN - 2: ZUR VALIDIERUNG - 3: VALIDIERT - 4: AUSGESTELLT - 5: BENACHRICHTIGT - 6: ABGELEHNT - 7: ANNULLIERT - 8: ENDE HERUNTERLADEN
ES_NOMBRE	VARCHAR2 (50)	Name des Bearbeitungszustands	JA	
ES_DESCRIPCION	VARCHAR2 (150)	Beschreibung des Bearbeitungsstatus der Apostille	NEIN	

Tabelle der Länder

TC_PAIS				
Enthält die Daten eines Landes, die die Applikation zu einem bestimmten Zeitpunkt des Apostillierungsprozesses benötigt				
Spalte	**Datenart**	**Beschreibung**	**Obligatorisch**	**Werte**
PA_CLAVE	NUMBER (3)	Einmaliger Länderidentifikator	JA	
PA_NOMBRE	VARCHAR2 (100)	Name des Landes	JA	

Tabelle der Provinzen

TC_PROVINCIA				
Hier sind die Daten einer Provinz enthalten, die die Applikation zu einem bestimmten Zeitpunkt des Apostillierungsprozesses benötigt				
Spalte	**Datenart**	**Beschreibung**	**Obligatorisch**	**Werte**
PR_CLAVE	NUMBER (2)	Einmaliger Provinzidentifikator	JA	
PR_NOMBRE	VARCHAR2 (100)	Name der Provinz	JA	

Tabelle der Gemeinden

TC_MUNICIPIO				
Hier sind die Daten einer Gemeinde enthalten, die die Applikation zu einem bestimmten Zeitpunkt des Apostillierungsprozesses benötigt				
Spalte	**Datenart**	**Beschreibung**	**Obligatorisch**	**Werte**
MU_CLAVE	NUMBER (4)	Einmaliger Gemeindeidentifikator	JA	
MU_NOMBRE	VARCHAR2 (100)	Name der Gemeinde	JA	
MU_PR_CF_PROV	NUMBER (2)	Einmaliger Identifikator der Provinz, der die Gemeinde angehört.	JA	