

**INSTITUTO TECNOLÓGICO DE TOLUCA**

**INFORME TÉCNICO DE RESIDENCIA PROFESIONAL**

**“Sistema de extracción y conversión de información curricular, almacenada en formato PDF, a formatos XML y JSON”**

**Ingeniería en Sistemas Computacionales**

**PRESENTA:**

**Lara Madero Axel**

**No. CONTROL:**

**19280766**

**ASESOR INTERNO:**

**MADSI. Paola Saray Monterrosas Cabrera**

**ASESOR EXTERNO:**

**Dr. Federico del Razo López**

**METEPEC, ESTADO DE MÉXICO, JULIO DE 2025**

# Agradecimientos

# Resumen

# Índice

[1. Agradecimientos 2](#_Toc198218815)

[2. Resumen 2](#_Toc198218816)

[3. Índice 3](#_Toc198218817)

[4. Índice de ilustraciones 6](#_Toc198218818)

[5. Índice de Tablas 7](#_Toc198218819)

[6. Introducción 8](#_Toc198218820)

[7. Descripción de la empresa 8](#_Toc198218821)

[8. Planteamiento del problema 9](#_Toc198218822)

[9. Objetivos 10](#_Toc198218823)

[9.1 Objetivo general 10](#_Toc198218824)

[9.2 Objetivos específicos 10](#_Toc198218825)

[10. Justificación 11](#_Toc198218826)

[10.1 Impacto del proyecto 11](#_Toc198218827)

[10.1.1 A corto plazo 11](#_Toc198218828)

[10.1.2 A mediano plazo 11](#_Toc198218829)

[10.1.3 A largo plazo 11](#_Toc198218830)

[11. Marco teórico 13](#_Toc198218831)

[11.1 Fundamentos de la estructuración y validación de XML 13](#_Toc198218832)

[11.1.1 Estructura de datos en XML (Extensible Markup Lenguage) 13](#_Toc198218833)

[11.1.2 Validación de XML 14](#_Toc198218834)

[11.2 Esquemas de validación en XML 14](#_Toc198218835)

[11.2.1 Document Type Definition (DTD) 14](#_Toc198218836)

[11.2.2 XML Schema Definition (XSD) 15](#_Toc198218837)

[11.2.3 Web Ontology Lenguage (OWL) 16](#_Toc198218838)

[11.3 Análisis de tecnologías para la construcción del sistema 16](#_Toc198218839)

[11.3.1 Frameworks para el desarrollo de APIs en Python (FastApi) 16](#_Toc198218840)

[11.3.2 Manipulación y validación de XML (xml.etree) 17](#_Toc198218841)

[11.3.3 Validación de XML con DTD y XSD (lxml) 17](#_Toc198218842)

[11.3.4 Conversión de XML a otros formatos (xmltodict) 18](#_Toc198218843)

[11.3.5 Procesamiento semántico de XML con OWL y RDF 18](#_Toc198218844)

[11.4 Tecnologías para el desarrollo del cliente web 18](#_Toc198218845)

[11.4.1 React.js 18](#_Toc198218846)

[11.4.2 Next.js 19](#_Toc198218847)

[12. Procedimiento y descripción de las actividades realizadas 20](#_Toc198218848)

[12.1 Metodología aplicada 20](#_Toc198218849)

[12.2 Análisis de requerimientos 22](#_Toc198218850)

[12.2.1 Análisis de la estructura y datos exportados desde SECIHTI 22](#_Toc198218851)

[12.2.2 Identificación de Requerimientos de PRODEP y TecNM 27](#_Toc198218852)

[12.2.3 Definición de criterios de homologación 27](#_Toc198218853)

[12.3 Diseño de los esquemas XML y OWL 28](#_Toc198218854)

[12.3.1 Diseño del esquema DTD y XSD para la estructuración estandarizada de datos curriculares 28](#_Toc198218855)

[12.3.2 Creación de la ontología OWL para el modelado semántico de la información curricular 31](#_Toc198218856)

[12.3.3 Definición de las reglas de transformación para la interoperabilidad entre plataformas 34](#_Toc198218857)

[12.4 Desarrollo del módulo de transformación 34](#_Toc198218858)

[12.5 Consultas XQuery/XPath 34](#_Toc198218859)

[12.6 Aplicativo complementario para validación y gestión de datos curriculares 34](#_Toc198218860)

[12.6.1 Objetivos del aplicativo 34](#_Toc198218861)

[12.6.2 Arquitectura general del aplicativo 34](#_Toc198218862)

[12.6.3 Funcionalidades backend 35](#_Toc198218863)

[12.6.4 Funcionalidades frontend 36](#_Toc198218864)

[12.6.5 Flujo de operación y simulación 36](#_Toc198218865)

[13. Fuentes de información 38](#_Toc198218866)

# Índice de ilustraciones

[Ilustración 1. Diagrama del modelo incremental 20](#_Toc198218867)

[Ilustración 2. Información personal DTD 28](#_Toc198218868)

[Ilustración 3. Información Personal XSD 30](#_Toc198218869)

[Ilustración 4. Información personal OWL 33](#_Toc198218870)

# Índice de Tablas

[Tabla 1. DTD vs XSD 15](#_Toc198218871)

[Tabla 2. Características del flujo de trabajo 21](#_Toc198218872)

# Introducción

# Descripción de la empresa

El Instituto Tecnológico de Toluca ubicado en Metepec, Estado de México cuya dirección es Av. Tecnológico 100-s/n, Agrícola, 52149. Es una institución mexicana de educación superior pública reconocida por su compromiso con la excelencia académica, la innovación tecnológica y la formación de profesionistas altamente capacitados en las áreas industrial y de servicios. Promueve el desarrollo integral de sus estudiantes mediante una educación basada en valores, contribuyendo al progreso regional a través de la investigación, la capacitación y la actualización profesional. Con una visión sostenible y equitativa, el Tecnológico trabaja en estrecha colaboración con los sectores productivos y educativos para impulsar el desarrollo social, económico y cultural de la comunidad.

# Planteamiento del problema

En el ámbito académico, los docentes frecuentemente deben presentar sus currículos para diversos procesos administrativos ante instituciones como el Programa para el Desarrollo Profesional Docente (PRODEP), el Tecnológico Nacional de México (TecNM) o la Secretaría de Ciencias, Humanidades, Tecnología e Innovación (SECIHTI).

Aunque los formatos requeridos por estas instituciones comparten información similar, cada uno exige un formato específico y datos estructurados de manera distinta.

Dicha situación, implica que los docentes deban dedicar tiempo considerable a adaptar manualmente su información curricular para cumplir con las exigencias de cada institución. Esto no solo lo vuelve un proceso repetitivo, sino que incrementa la posibilidad de errores, inconsistencias y pérdida de información relevante. La ausencia de un sistema automatizado que permita la extracción y transformación de datos a formatos estandarizados dificulta la optimización de este proceso.

Actualmente, no se cuenta con una herramienta que facilite la gestión eficiente de los datos curriculares a través de una estructura reutilizable y compatible con los requerimientos de múltiples instituciones. Esta carencia impacta negativamente en el tiempo y esfuerzo invertido por los docentes.

# Objetivos

## Objetivo general

Desarrollar un modelo de estructuración y validación de datos curriculares mediante la creación de esquemas XML usando DTD, XSD. Así como una ontología OWL con el propósito de estandarizar la información y facilitar la futura interoperabilidad con los CVU (Currículum Vitae Únicos) de SECIHTI, PRODEP y TecNM.

## Objetivos específicos

* Diseñar un esquema XML (DTD,XSD) y una ontología OWL que defina la estructura estándar para la información curricular, compatible con los requerimientos de instituciones académicas como PRODEP, TecNM y SECIHTI.
* Desarrollar un proceso de transformación y homologación de datos curriculares, tomando como base los archivos XML exportados desde SECIHTI, para facilitar su correcta conversión y compatibilidad con los sistemas de PRODEP y TecNM.
* Desarrollar un proceso de validación y transformación de datos que convierta la información extraída en archivos XML conformes a XML-Schemas en archivos JSON estructurados, facilitando la integridad, consistencia y compatibilidad con los sistemas de destino.
* Implementar consultas y scripts tanto con XQuery como con XPath para verificar la correcta estructuración, transformación de los datos curriculares en XML, garantizando su conformidad con los esquemas DTD/XSD y facilitando su interoperabilidad con los sistemas PRODEP y TecNM.
* **Explorar la viabilidad de una aplicación complementaria** que facilite la visualización, conversión y descarga de la información curricular en formatos XML y JSON.
* Realizar pruebas tanto funcionales como de rendimiento del sistema para asegurar la precisión en la extracción, validación y conversión de los datos curriculares.
* Generar documentación técnica para futuras adaptaciones.

# Justificación

Actualmente, los docentes deben capturar y actualizar su información curricular en múltiples plataformas institucionales; tales como, SECIHTI, PRODEP y TecNM. Este proceso repetitivo; consume tiempo considerable, además de que puede generar inconsistencias en la información por errores de captura, omisiones o diferencias en los formatos exigidos por cada sistema.

**Para resolver lo anterior, se propone el desarrollo de un modelo de estructuración, validación y transformación de datos curriculares, basado en esquemas XML (DTD, XSD) junto con una ontología OWL. Esto permitirá homologar la información curricular y facilitar su interoperatividad entre los sistemas ya mencionados.**

**La plataforma SECIHTI será la fuente principal de la información curricular. Esto mediante la exportación de datos, se buscará estandarizar la estructura de los CVU para facilitar su integración en PRODEP y TecNM. La validación mediante esquemas como con la ontología asegurará la consistencia y compatibilidad de los datos, reduciendo la duplicidad de esfuerzos y errores de captura.**

## Impacto del proyecto

### A corto plazo

Los docentes contarán con una herramienta funcional que reducirá significativamente el tiempo invertido en la preparación de sus currículos. Esto facilitará la entrega puntual y sin errores a las instituciones académicas.

### A mediano plazo

La Institución beneficiará a su personal docente al dotarlos de un sistema tecnológico que eleve la eficiencia administrativa y promueva el cumplimiento de estándares curriculares. Además, esto fomentará la adopción de buenas prácticas tecnológicas en otros procesos similares.

### A largo plazo

Se establecerá un precedente para el desarrollo de soluciones tecnológicas que integren procesos administrativos y académicos, contribuyendo al avance de la digitalización en el sector educativo. Esto también posicionará al Instituto Tecnológico de Toluca como un referente en innovación y soporte a la comunidad académica.

El proyecto destaca por su utilidad práctica y su enfoque innovador, ya que aprovecha tecnologías como XML-Schemas, XQuery, XPath y Python. Estas herramientas no solo garantizan la precisión y flexibilidad del sistema, sino que también aseguran su escalabilidad para futuros requerimientos. En términos económicos, el sistema reducirá los costos asociados con errores manuales y tiempos improductivos. Desde una perspectiva social y administrativa, mejorará la calidad y la consistencia de la información presentada por los docentes.

En conclusión, este proyecto **responde a la necesidad de reducir el esfuerzo manual en la gestión de información curricular, mejorando la precisión, confiabilidad y automatización del proceso de integración de CVU en las distintas plataformas académicas.**

# Marco teórico

El presente marco teórico establece los fundamentos conceptuales y tecnológicos que sustentan el desarrollo de un sistema de extracción y conversión de datos curriculares a formatos XML. Este proyecto responde a la necesidad de optimizar la gestión de información curricular en instituciones académicas, garantizando su estandarización y compatibilidad con los formatos exigidos por distintos organismos.

En este contexto, el presente capítulo aborda los principios fundamentales de la estructuración y validación de XML, explorando sus aplicaciones en el ámbito curricular. Se analizan los distintos esquemas de validación disponibles, destacando sus tanto sus ventajas como sus limitaciones en el procesamiento de datos. Adicionalmente, se examinan las tecnologías clave para la implementación del sistema, incluyendo frameworks de desarrollo en Python como FastAPI, herramientas para la manipulación de XML y metodologías para su integración con plataformas web mediante React y Next.js.

## Fundamentos de la estructuración y validación de XML

### Estructura de datos en XML (Extensible Markup Lenguage)

XML es un lenguaje de marcado cuyo propósito es almacenar y transportar datos, caracterizado por su legibilidad tanto en personas como en máquinas permite que sea una herramienta versátil para el intercambio de información entre sistemas (Diego Lázaro, 2018). Todo documento XML está compuesto de estructuras lógicas y físicas, mientras las estructuras lógicas se representan mediante declaraciones, elementos o comentarios; por otro lado, las físicas se organizan mediante unidades que almacenan los datos (Hernández, 2014).

Un documento bien formado debe cumplir con una serie de reglas sintácticas básicas y un correcto anidamiento de elementos, esta estructura jerárquica del XML resulta especialmente relevante para la estandarización de información curricular docente; permite organizar de manera lógica las diversas secciones del currículum vitae. Esto facilita la representación clara y estructurada de la información.

### Validación de XML

La validación de un documento XML es un proceso que consiste en verificar si cumple con las reglas estructurales de contenido y de tipo de dato definidas por un esquema específico (Lledó, 2010); este proceso garantiza tanto la corrección como la coherencia en los datos. Para este proyecto, se hace énfasis en dos principales niveles de validación:

* **Validar el “bien formado”**

Es un nivel básico de validación centrado en que un documento XML cumpla con la presencia de un único elemento raíz, el correcto cierre de las etiquetas, el anidamiento adecuado de elementos y la codificación correcta de caracteres especiales (Sánchez, 2025). Un documento sintácticamente correcto puede ser procesado y analizado fácilmente.

* **Validación con esquemas**

Siendo un nivel más avanzado de validación, verifica que el documento XML se ajuste a una estructura y reglas definidas; estos archivos pueden ser un DTD o XSD y especifican los elementos y/o atributos permitidos, así como su jerarquía, tipo de dato que pueden contener u otras restricciones (FasterCapital, 2024). Usar esquemas asegura una integridad y calidad en los datos; así como un documento con información válida acorde a las reglas.

Al garantizar que los datos intercambiados cumplan con estándares acordados se promueve la interoperabilidad entre sistemas y reduce las posibilidades de corrupción de datos; especialmente si la información curricular exige un formato específico por institución o simplemente cierta información se presta a una mala interpretación en los distintos sistemas.

## Esquemas de validación en XML

### Document Type Definition (DTD)

Es el formato de esquema original y más antiguo utilizado para la validación de documentos XML; heredado del lenguaje SGML (Diego Lázaro, 2018), define la gramática que un documento XML debe tener para ser considerado válido (*DTD y XML Schema.pdf*, s/f) de igual modo, su sintaxis simple y concisa permiten la definición de entidad (Microsoft, 2024). Estos documentos pueden definirse dentro del XML o en un archivo separado con extensión *dtd.*

Dentro del proyecto, el uso de DTD facilita la creación y mantenimiento de estructuras XML, así como la compatibilidad con software existente; sin embargo, el soporte para tipos de datos en un DTD es bastante limitado lo cual genera conflictos si se intenta combinar información curricular proveniente de diferentes fuentes o instituciones. En general, DTD se considera menos potente y expresivo que alternativas como XSD para describir aspectos formales de un documento XML.

### XML Schema Definition (XSD)

Presentado como la evolución de DTD y recomendada por el World Wide Web Consortium (W3C); constituye el lenguaje de esquema más complejo y potente para validar documentos XML (Diego Lázaro, 2018), ofrece un soporte robusto para una aplia variedad de tipos de datos predefinidos tales como numéricos, cadenas de texto, booleanos y fechas (IBM, 2021). Además, XSD permite la definición de restricciones (facets) como rangos, patrones o longitudes (*DTD y XML Schema.pdf*, s/f).

Los esquemas XSD ofrecen la capacidad de definir con precisión tanto la estructura como los tipos de datos de los diferentes campos; incluyendo formatos específicos para fechas, números y otros tipos de información; asimismo, proporciona un mayor control sobre la obligatoriedad y la repetición de los diferentes campos. Pare tener una visión clara de las diferencias entre DTD y XSD en la validación de información se puede visualizar la *Tabla 1. DTD vs XSD.*

Tabla 1. DTD vs XSD

|  |  |  |
| --- | --- | --- |
| Característica | DTD | XSD |
| *Sintaxis* | No basada en XML | Basada en XML |
| *Tipos de datos* | Limitados (#PCDATA) | Extensos y personalizables |
| *Espacios de nombres* | No soporta | Soporta |
| *Cardinalidad* | Control limitado | Control preciso (minOccurs, maxOccurs) |
| *Extensibilidad* | No extensible | Extensible |
| *Legibilidad* | Generalmente más legible | Puede ser complejo |
| *Potencia expresiva* | Menor | Mayor |

### Web Ontology Lenguage (OWL)

Es un lenguaje de marcado semántico concebido para la publicación y el intercambio de ontologías en la World Wide Web; su desarrollo surge como una extensión del vocabulario de RDF (Resource Description Framework), derivado a su vez del lenguaje DAML+OIL Web Ontology (W3C, 2025). Su propósito fundamental radica en optimizar el procesamiento del contenido informativo por parte de las aplicaciones, superando las capacidades de XML, RDF y RDF Schema; para ello, proporciona un vocabulario más amplio acompañado de una semántica formal (*Introduction to OWL.pdf*, s/f).

Una ontología OWL puede establecer un lenguaje común y formalizado para datos curriculares, estandariza el significado de términos clave, lo cual facilita que las diferentes plataformas interpreten la información curricular de manera consistente; de igual modo, permite definir relaciones semánticas explícitas entre diversos elementos, lo cual proporciona un contexto semántico rico a los datos curriculares.

## Análisis de tecnologías para la construcción del sistema

### Frameworks para el desarrollo de APIs en Python (FastApi)

FastApi es un framework web moderno y de alto rendimiento para la construcción de API’s con Python, destacando por su velocidad comparable a la de NodeJS y Go gracias a su base en Starlette y Pydantic (FastAPI, 2025). Su diseño intuitivo agiliza el desarrollo al reducir errores y optimizar la experiencia al codificar, todo esto junto a su documentación interactiva automática basada en OpenAPI y JSON Schema mejora la comprensión y prueba de la API (FastAPI, 2025).

Para este proyecto; el uso de FastAPI facilita la creación de endpoints permitiendo operaciones de creación, lectura, actualización y eliminación. El uso de Python, hace más sencillo el trabajo con documentos XML y sus validaciones mediante transformaciones o lecturas. El uso de sus distintas librerías nativas que contiene el lenguaje lo vuelve la herramienta más indicada para este caso.

### Manipulación y validación de XML (xml.etree)

El módulo *xml.etree.ElementTree* de Python proporciona una API sencilla y eficiente que nos permite analizar y generar datos en formato XML, representa los documentos como árboles de elementos mediante las clases *ElemetnTree* y *Element* (Python, 2025; xml.etree.ElementTree, 2019)*.* Permite procesar XML desde archivos o cadenas de texto mediante sus funciones *ET.parse()* y *ET.fromstring()*, esto facilita la navegación y manipulación de su estructura mediante iteraciones tanto sobre los elementos como en etiquetas; además de que ofrece soporte básico para XPath en la búsqueda de elementos o funciones que permitan modificar el contenido XML (Python, 2025).

En el contexto del proyecto, la librería será utilizada para analizar los archivos XML mediante el acceso a los datos específicos que se encuentran dentro de los diferentes elementos y atributos; de igual modo, para recuperar archivos dentro de las funciones de FastAPI que vayan a ser utilizados en otras clases.

### Validación de XML con DTD y XSD (lxml)

La librería lxml de Python permite el procesamiento de documentos XML y HTML, se destaca por combinar la velocidad de bibliotecas en lenguaje C (*libxml2* y *libxslt*) con la librería *xml.etree*; sin embargo, contiene un soporte completo para XPath 1.0, lo que permite realizar consultas complejas y eficientes dentro de los documentos XML (ScrapingAnt, 2024).

Esta librería se convierte en una herramienta fundamental para la validación de archivos XML que contengan la información curricular, permitirá verificar que estos archivos cumplen con los esquemas DTD y XSD definidos. Su soporte para XSD en particular, será de gran valor dada la potencia y flexibilidad de este lenguaje.

### Conversión de XML a otros formatos (xmltodict)

El módulo xmltodict de Python fue diseñado para facilitar el trabajo con datos XML al permitir su conversión a estructuras de datos nativas del lenguaje; específicamente diccionarios (Maciek, 2024). Usar este módulo permite tomar un archivo XML y transformarlo a distintos formatos separando las etiquetas y los valores de los elementos o atributos (DigitalOcean, 2022).

La librería es de gran utilidad para transformar los datos curriculares en formato XML a una estructura de diccionario de Python, lo cual resulta en una sencilla manipulación y futura transformación a otros formatos como JSON o RDF para la interacción con el cliente web de la aplicación y con FastAPI para la validación de información.

### Procesamiento semántico de XML con OWL y RDF

El OWL construido sobre la base del RDF permite añadir una capa de semántica a los datos que se encuentran estructurados en el documento XML, el modelo de datos RDF representa información sobre los recursos disponibles mediante tripletas sujeto-predicado-objeto (Martínez, 2020). El OWL extiende las capacidades del RDF proporcionando un vocabulario más rico y una semántica formal para la descripción de ontologías, incluyendo la definición de las clases, propiedades y relaciones entre ellas (Semantic Web Technologies, 2025).

El uso de RDF permitirá validar la ontología OWL mediante Python, esto gracias al modelo estructurado que facilita el análisis de los datos. Esta validación permitirá verificar que la ontología cumpla con las reglas establecidas, facilitando su integración y uso en aplicaciones semánticas; así como su interoperabilidad con otros sistemas.

## Tecnologías para el desarrollo del cliente web

### React.js

React.js es una biblioteca de JavaScript de código abierto diseñada para desarrollar interfaces de usuario de forma eficiente; su arquitectura basada en componentes permite dividir la UI en elementos reutilizables e independientes, facilitando el mantenimiento y la escalabilidad (React, 2024). Además, optimiza el rendimiento mediante un **Virtual DOM**, actualizando solo los elementos modificados en lugar de renderizar toda la interfaz (Deyimar, 2020).

React.js será utilizado para construir el cliente web, permitiendo a los docentes interactuar de manera intuitiva con su información curricular y adaptarla a los requerimientos de las distintas plataformas; esto permite crear componentes reutilizables para la visualización de cada uno de los campos. El manejo del estado de la interfaz permitirá reflejar en tiempo real los cambios realizados por los docentes.

### Next.js

Next.js es un framework de React que amplía sus capacidades con renderizado del lado del servidor (**SSR**) y generación de sitios estáticos (**SSG**), mejorando el SEO y el rendimiento inicial al entregar páginas preprocesadas en el servidor o durante la construcción (Next.js, 2025). Facilita el enrutamiento con un sistema basado en la estructura de archivos dentro del directorio **pages** y permite la creación de **API Routes** dentro del proyecto, integrando lógica del lado del cliente con servicios como FastAPI (aplyca, 2022).

El uso de Next.js permite complementar a React.js al mejorar el rendimiento de la aplicación web para los docentes, gracias al SSR y SSG permite una experiencia de usuario más fluida; de igual modo, simplificaría la gestión de las diferentes secciones de la interfaz, esto junto a las API Routes podrán utilizarse para implementar cierta lógica del lado del cliente que interactúe directamente con FastApi para la manipulación y obtención de datos.

# Procedimiento y descripción de las actividades realizadas

## Metodología aplicada

Para el desarrollo del "Sistema de extracción y conversión de información curricular a formatos XML y JSON", se adoptó la metodología incremental, una variante del modelo evolutivo que permite construir el sistema de forma progresiva mediante entregas parciales con funcionalidades completas; esta elección se fundamenta en la naturaleza dinámica del proyecto, la cual requiere flexibilidad para incorporar ajustes o refinamientos en los requisitos y componentes a medida que avanza el análisis, incluso en etapas ya avanzadas.

El modelo incremental se caracteriza por dividir el ciclo de vida del desarrollo en una serie de incrementos, cada incremento construye una porción funcional del sistema lo cual permite una evolución progresiva y adaptación a cambios de manera más eficiente. Esto último es particularmente ventajoso para el proyecto, dado que la comprensión detallada de los formatos curriculares junto con la definición de esquemas y ontologías evolucionan a medida que se avanza.

El desarrollo del sistema se abordará mediante una secuencia de incrementos, donde cada uno sigue un ciclo similar adaptado a sus objetivos específicos tal como se muestra en la *Ilustración 1. Diagrama del modelo incremental.*

Ilustración . Diagrama del modelo incremental

Cada incremento se enfoca en un conjunto específico de funcionalidades o componentes del sistema, partiendo de los requisitos generales del proyecto. Las principales características de cada incremento y cómo se gestiona el flujo de trabajo se resumen en la *Tabla 2. Características del flujo de trabajo*.

Tabla . Características del flujo de trabajo

|  |  |  |
| --- | --- | --- |
| Fase del incremento | Descripción de actividades | Gestión de cambios |
| Planificación del incremento | Definición de objetivos, alcance y entregables para el incremento | Los nuevos requisitos o cambios se analizan |
| Análisis y diseño | Diseño detallado de los componentes a desarrollar en el incremento | El diseño permite la integración con incrementos futuros |
| Desarrollo | Codificación de los componentes definidos siguiendo las especificaciones del diseño | El uso de un código modular facilita futuras modificaciones o extensiones |
| Pruebas y validación | Verificación funcional de los componentes desarrollados en el incremento | Los errores se corrigen dentro del mismo incremento, en caso de un cambio mayor se planifica su impacto |
| Revisión e integración | Presentación del incremento funcional permitiendo integrarlo con incrementos anteriores | La retroalimentación puede generar ajustes en incrementos pasados o futuros |
| Evolución del sistema | El sistema se construye progresivamente con cada incremento | La "susceptibilidad a cambios en puntos ya realizados" se manejan como requisitos para nuevos ciclos de refinamiento |

El uso de este modelo permite desarrollar un sistema robusto y adaptado a los requisitos finales, gestionando la complejidad junto con los cambios inherentes a un proyecto de esta naturaleza.

## Análisis de requerimientos

### Análisis de la estructura y datos exportados desde SECIHTI

El objetivo primordial de esta fase es realizar un análisis de la estructura de datos y los elementos de información curricular gestionados por la SECIHTI a través de su plataforma Rizoma; este entendimiento es crucial, ya que la información proveniente de dicha institución constituye la fuente primaria para el sistema de extracción y conversión propuesto. El análisis busca identificar la semántica, granularidad, tipos de datos, restricciones y relaciones entre los diferentes campos curriculares para fundamentar el diseño de los esquemas XML normalizados y la ontología OWL.

**Fase preliminar**

En un inicio, se llevó a cabo una revisión preliminar utilizando un CVU de ejemplo; esto permitió identificar un conjunto básico de datos personales comúnmente requeridos para los docentes, tales como:

* Clave Única de Registro de Población (CURP)
* Registro Federal de Contribuyentes (RFC)
* Sexo
* Domicilio.
* Fecha de nacimiento.
* País de nacimiento.
* Nacionalidad.
* Estado civil

Esta fase proporcionó una visión general, permitió reconocer la necesidad de un análisis más detallado y directo de la estructura subyacente en la plataforma de Rizoma.

**Análisis actual de la plataforma Rizoma**

El análisis se centra en la estructura de la información solicitada directamente en el portal Rizoma de SECIHTI; mediante un proceso iterativo se explora la amplitud y profundidad de los datos curriculares. A continuación, se detallan los hallazgos obtenidos y organizados según los apartados identificados en la plataforma:

1. **Apartado “Mi perfil”**

Este apartado agrupa la información general y de contacto del usuario docente. Los elementos identificados son:

1. **Semblanza:**

* Descripción: Campo alfanumérico destinado a una breve descripción del perfil profesional del docente.
* Cardinalidad: Obligatorio.
* Restricciones: Longitud máxima de 2000 caracteres.

1. **Contacto:**

* Descripción: Agrupa diversos medios de comunicación del docente.
* Cardinalidad: Obligatorio (debe contener al menos uno de los subcampos).
* Subcampos (opcionales, pero al menos uno es requerido):
  + Correo principal: Alfanumérico, debe contener el carácter "@".
  + Móvil principal: Numérico, 10 dígitos.
  + Teléfono principal: Numérico, 10 dígitos.
  + Red social: Alfanumérico, opcional.

1. **Habilidades:**

* Descripción: Permite registrar las competencias del docente.
* Cardinalidad: Opcional (0 a N habilidades).
* Atributos: Cada habilidad posee un atributo obligatorio denominado "experiencia", con valores predefinidos (25, 50, 75, 100) que representan un porcentaje.

1. **Área de conocimiento:**

* Descripción: Clasifica la especialización del docente.
* Cardinalidad: Obligatorio.
* Componentes (todos obligatorios y alfanuméricos): Área, Campo, Disciplina, Subdisciplina.

1. **Identificadores de autor:**

* Descripción: Registra identificadores de publicaciones o perfiles académicos.
* Cardinalidad: Opcional (0 a N identificadores).
* Atributos: Cada identificador tiene un atributo obligatorio "organización" con valores predefinidos (ORCID, Thomson Reuters ResearcherID, arXiv ID, PubMed ID, OpenID). (Nota: Se asumió "ORC" como ORCID y "Open" como OpenID, verificar nombres exactos en la plataforma).

1. **Información general:**

* Descripción: Datos demográficos y de identificación fundamentales del docente.
* Cardinalidad: Obligatorio.
* Campos (todos obligatorios, salvo especificación):
  + CURP: Alfanumérico, 18 caracteres.
  + Nombre(s): Texto.
  + Primer Apellido: Texto.
  + Segundo Apellido: Texto (se asume obligatorio aunque no lo indique explícitamente el input, es práctica común en México).
  + Fecha de nacimiento: Fecha, formato AAAA-MM-DD.
  + Sexo: Texto, valores ("masculino", "femenino").
  + País de nacimiento: Texto.
  + Entidad federativa: Texto (aplicable si País de nacimiento es México).
  + RFC: Alfanumérico, 13 caracteres.
  + Estado civil: Texto, conjunto de valores predefinidos (ej. "Casado(a)", "Soltero(a)", etc.).
  + Nacionalidad: Texto.

1. **Domicilio:**

* Descripción: Información detallada de la residencia del docente.
* Cardinalidad: Obligatorio.
* Componentes Principales:
  + Ubicación: Texto, obligatorio, valores ("México", "Extranjero").
  + Si Ubicación es "México":
    - Código postal: Numérico, 5 dígitos, obligatorio.
    - Estado, Municipio o alcaldía, Localidad: Campos de texto obligatorios.
    - Asentamiento, Tipo de asentamiento: Campos alfanuméricos opcionales.
    - Información extra: Booleano (Verdadero/Falso), se asume que es un atributo.
    - Tipo (domicilio): Texto, obligatorio, valores ("Urbano", "Rural").
      * Subestructura para *Urbano* (Tipo de vialidad, Nombre de la vialidad obligatorios; Número exterior, Número interior opcionales con sub-atributos).
      * Subestructura para *Rural* (Tipo de vialidad, Nombre de la vialidad obligatorios; varios campos opcionales como Administración, Derecho de tránsito, Código, etc., y Número exterior/interior opcionales con sub-atributos).
      * Subestructura para *Urbano* (Tipo de vialidad, Nombre de la vialidad obligatorios; Número exterior, Número interior opcionales con sub-atributos).
      * Subestructura para *Rural* (Tipo de vialidad, Nombre de la vialidad obligatorios; varios campos opcionales como Administración, Derecho de tránsito, Código, etc., y Número exterior/interior opcionales con sub-atributos).
    - *Si Información extra es Verdadero:*
      * Referencias: Opcional (0 a 3 ocurrencias), con atributos "Tipo de calle" y "Nombre de la calle".
      * Descripción: Alfanumérico, opcional, hasta 500 caracteres.
  + Si Ubicación es “Extranjero”:
    - País, Código postal, Estado, Ciudad, Condado, Calle, Número, Descripción: Campos alfanuméricos obligatorios (con restricciones específicas para Descripción).

1. **Contactos de emergencia:**

* Descripción: Personas a contactar en caso de emergencia.
* Cardinalidad: Opcional (0 a N contactos).
* Campos por contacto (mayoría obligatorios): Nombre, Primer apellido, Parentesco, Fecha de nacimiento (DD-MM-AAAA), Sexo (Femenino, Masculino, No binario, Otro), Correo electrónico (con "@"), Lada, Teléfono, Móvil, Medio preferente, Ubicación (México/Extranjero). El Segundo apellido es opcional.
* Subestructura para Ubicación (México/Extranjero) similar a la del domicilio del docente, pero con campos específicos para contacto de emergencia.

1. **Dependientes económicos:**

* Descripción: Información sobre dependientes económicos del docente.
* Cardinalidad: Opcional (0 a N dependientes).
* Atributo principal: "Nacionalidad del dependiente" (Mexican/Foreign).
* Campos (mayoría obligatorios, con variaciones si es Mexicano o Extranjero):
  + Si Mexican: CURP, Nombre, Apellidos, Fecha de nacimiento (DD-MM-AAAA), Edad, Parentesco (valores predefinidos), Sexo (valores predefinidos), Orden.
  + Si Foreign: Nombre, Apellidos, Fecha de nacimiento (DD-MM-AAAA), Edad, Parentesco, Sexo, Orden.

1. **Apartado “Mi trayectoria”**

Este apartado registra la formación académica formal del docente. Puede contener múltiples registros.

* Cardinalidad del Apartado: Opcional (0 a N registros de trayectoria).
* Campos por registro (todos obligatorios):
  + Nivel de escolaridad: Texto, valores predefinidos (Doctorado, Especialidad, Licenciatura, etc.).
  + Título: Alfanumérico.
  + Estado del grado: Texto, valores predefinidos (Créditos terminados, Grado obtenido, etc.).
  + Institución
    - Atributos obligatorios: "Tipo de institución" (Extranjera/Nacional), "Sector" (Pública/Privada).

### Identificación de Requerimientos de PRODEP y TecNM

### Definición de criterios de homologación

## Diseño de los esquemas XML y OWL

### Diseño del esquema DTD y XSD para la estructuración estandarizada de datos curriculares

Estos esquemas son fundamentales para validar la consistencia e interoperabilidad de la información curricular que el sistema procesará; para este proyecto se contempla el diseño y uso tanto de DTD como de XSD, aprovechando las capacidades de cada uno.

El diseño de los esquemas DTD y XSD se basa en los hallazgos del punto **“11.2.1 Análisis de la estructura y datos exportados desde SECIHTI”**. El proceso es iterativo:

1. *Identificar entidades y elementos*
2. *Definir las estructuras jerárquicas*
3. *Especificación de tipos de datos y restricciones*
4. *Desarrollo iterativo*

**Diseño del DTD**

El DTD se utiliza para definir la estructura gramatical básica del documento XML del CVU; establece los elementos permitidos, su orden y anidamiento. A continuación, se presenta un fragmento del DTD diseñado para la sección de datos personales, como ejemplo inicial *(Ilustración 2. Información personal DTD)*:

Ilustración . Información personal DTD

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Este DTD declara el elemento raíz **cvu** que contiene **personalData**, el cual a su vez engloba los elementos básicos de identificación. Si bien el DTD es eficaz para esta definición estructural, presenta limitaciones en la especificación detallada de los tipos de datos y restricciones complejas, campos en los que XSD ofrece mayores ventajas. El atributo **xsi:noNamespaceSchemaLocation** se incluye para permitir la vinculación con un esquema XSD para una validación más robusta.

**Diseño del XSD**

Para una validación más exhaustiva y precisa, se ha desarrollado un esquema XSD. XSD permite definir con mayor rigor los tipos de datos, aplicar patrones (expresiones regulares) para formatos específicos, establecer enumeraciones de valores permitidos y definir tipos complejos reutilizables.

Tomando como base la misma sección de datos personales, podemos visualizar su estructura en la *Ilustración 3. Información personal XSD*.

En dicho esquema podemos resaltar los siguientes puntos:

* Se define un elemento raíz **cvu** que contendrá todas las secciones del currículum
* La sección **personalData** que agrupa los datos de identificación
* Se utilizan tipos de datos específicos como **xs:date** para **fechaNacimiento**
* Se definen **simpleType** personalizados (*CURPType* y *RFCType)* que utilizan **xs:pattern** para validar el formato exacto de la CURP y el RFC.
* Para campos como **Sexo** y **EstadoCivil**, se utilizan **xs:enumeration** para restringir los valores a un conjunto predefinido, asegurando la consistencia.
* El elemento **Domicilio** se muestra con **type="xs:string"** como un marcador de posición inicial.

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración . Información Personal XSD

### Creación de la ontología OWL para el modelado semántico de la información curricular

Complementario al diseño de los esquemas XSD y DTD, que definen la estructura sintáctica de los datos curriculares en XML, se procede a la creación de una ontología utilizando OWL; el propósito fundamental de esta ontología es proporcionar un modelo semántico formal y explícito del dominio curricular. Este modelo enriquecerá la representación de los datos, permitiendo una mayor comprensión de las interrelaciones entre los conceptos, asegurando la consistencia semántica.

**Proceso de diseño de la ontología CVU**

El diseño de la ontología CVU se fundamenta en:

* El análisis detallado de la estructura de datos de SECIHTI-Rizoma (descrito en 11.2.1).
* Los criterios de homologación (definidos en 11.2.2) que buscan unificar la representación de la información.
* Los esquemas XML (DTD/XSD) desarrollados (descritos en 11.3.1), de los cuales la ontología toma la estructura base para enriquecerla semánticamente.

El desarrollo es un proceso iterativo con los siguientes pasos:

1. *Identificación de clases:* se definen las clases principales que representan los conceptos fundamentales.
2. *Definición de propiedades:* se establecen propiedades de datos para describir los atributos de las clases con valores literales y propiedades del objeto.
3. *Establecimiento de jerarquías y restricciones:* se organizan las clases en jerarquías y se aplican restricciones para refinar el significado.
4. *Población y refinamiento:* a medida que se incorporan más secciones la ontología se expande y refina.

**Componentes fundamentales**

Como punto de partida se ha modelado la sección correspondiente a la información personal del docente. El fragmento de ontología visualizado en la *Ilustración 4. Información personal OWL* ilustra los principales constructos OWL utilizados.

En dicho fragmento podemos resaltar lo siguiente:

* La definición de la clase **cvu:InformacionGeneral** permite agrupar los datos personales.
* El uso de **owl:DatatypeProperty** permite representar atributos con valores literales, especificando su dominio (**rdfs:domain**) y rango (**rdfs:range**).
* La creación de clases enumeradas mediante **owl:oneOf** permite definir conjuntos cerrados de valores permitidos.
* El uso de **owl:ObjectProperty** permite establecer relaciones entre una instancia de *cvu:InformacionGeneral* y una instancia de las clases enumeradas.
* La aplicación de restricciones de cardinalidad (**owl:qualifiedCardinality**) sobre la clase *cvu:InformacionGeneral* para asegurar, por ejemplo, que cada individuo tenga exactamente un valor para *cvu:tieneSexo*.
* Un ejemplo de restricción de rango de valores (**owl:allValuesFrom** con **xsd:maxInclusive**) en la clase *cvu:PersonaValida* para la propiedad *cvu:FechaNacimiento*.

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración . Información personal OWL

### Definición de las reglas de transformación para la interoperabilidad entre plataformas

## Desarrollo del módulo de transformación

## Consultas XQuery/XPath

## Aplicativo complementario para validación y gestión de datos curriculares

### Objetivos del aplicativo

El aplicativo complementario persigue los siguientes objetivos:

* Proveer una interfaz gráfica para la carga y validación de archivos XML curriculares.
* Demostrar el proceso de validación contra los esquemas DTD y XSD definidos en el servidor backend.
* Permitir la visualización estructurada de la información contenida en el XML validado.
* Facilitar la selección de datos específicos por parte del usuario para la generación de un nuevo archivo XML personalizado.
* Simular un flujo de trabajo donde un XML (representando la información de SECIHTI) es procesado, validado y transformado.

### Arquitectura general del aplicativo

El aplicativo se ha desarrollado bajo una arquitectura cliente-servidor:

* **Frontend (Cliente):** desarrollado con Next.js (un framework de React), proporciona la interfaz de usuario con la que el docente interactúa.
* **Backend (Servidor):** implementado con FastAPI (Python), expone una API RESTful que maneja la lógica de negocio, incluyendo la validación de los archivos XML y la generación de nuevos documentos.

La comunicación entre el frontend y el backend se realiza mediante peticiones HTTP, donde el frontend consume los servicios expuestos por la API del backend. El backend cuenta con la configuración de CORS (Cross-Origin Resource Sharing) para permitir las solicitudes desde el dominio del frontend.

### Funcionalidades backend

El servidor backend, implementado en Python con el framework FastAPI, ofrece los siguientes puntos de entrada (endpoints) principales:

1. **Endpoint de Carga y Validación de XML (/upload):**

* *Propósito:* recibir un archivo XML, validarlo contra los esquemas DTD y XSD definidos, y convertirlo a formato JSON si es válido.
* *Entrada:* un archivo XML (UploadFile) enviado desde el frontend.
* *Proceso (validation.py):*
  + El contenido del archivo XML se lee en memoria.
  + *Validación contra XSD:* se parsea el archivo XSD (Schema.xsd) y se valida el documento XML de entrada contra este esquema utilizando la librería **lxml**. Si hay errores, estos se retornan.
  + *Validación contra DTD:* Si la validación XSD es exitosa, se procede a validar el mismo documento XML contra el DTD (Schema.dtd) utilizando **lxml**.
  + *Conversión a JSON:* si ambas validaciones (XSD y DTD) son correctas, el contenido del XML se convierte a una estructura JSON utilizando la librería **xml.etree.ElementTree** para el parseo y una función personalizada para la conversión.
* *Salida:* un objeto JSON que indica el estado de la validación (valido: true) y los datos convertidos (data: json\_data), o una excepción HTTP con detalles del error en caso de fallo.

1. **Endpoint de Generación de XML (/xml\_gen):**

* *Propósito:* crear un nuevo archivo XML a partir de un conjunto de datos seleccionados por el usuario en el frontend.
* *Entrada:* un diccionario de Python (datos: dict) que contiene los elementos y valores a incluir en el nuevo XML.
* *Proceso:* se construye dinámicamente una estructura XML utilizando **xml.etree.ElementTree**. Se crea un elemento raíz <cvu> y una sección <personalData> donde se insertan los datos recibidos.
* *Salida:* una respuesta HTTP con el contenido del XML generado (application/xml).

### Funcionalidades frontend

El cliente web, desarrollado en Next.js, ofrece al usuario las siguientes capacidades:

1. **Carga de Archivo XML:** el usuario puede seleccionar un archivo XML de su sistema local a través de una interfaz de carga. Este archivo simula la información curricular que podría provenir de SECIHTI.
2. **Envío y Validación:** una vez cargado, el archivo se envía al endpoint /upload del backend FastAPI. El frontend espera la respuesta para conocer el resultado de la validación.
3. **Visualización de Datos:** si el XML es válido y procesado correctamente por el backend, la información (convertida a JSON) se retorna al frontend. Esta información se presenta al usuario de forma organizada, utilizando pestañas para separar las distintas secciones del currículum. Dentro de cada pestaña, los datos se muestran en tablas para facilitar su lectura y comprensión.
4. **Selección de Información:** el usuario tiene la capacidad de interactuar con las tablas de datos para seleccionar los campos o registros específicos que desea incluir en un nuevo archivo XML.
5. **Descarga de XML Personalizado:** tras la selección, el frontend recopila la información elegida y la envía al endpoint /xml\_gen del backend. Este último genera el nuevo XML con solo los datos seleccionados y el frontend facilita la descarga de este archivo al usuario.

### Flujo de operación y simulación

El flujo de trabajo implementado en el aplicativo simula un caso de uso central del proyecto:

1. Un usuario (docente) posee información curricular en un formato XML (este XML podría ser una exportación de SECIHTI o un archivo generado que se adhiere a la estructura esperada por el sistema).
2. El usuario carga este XML en el aplicativo frontend.
3. El backend valida la estructura y contenido del XML contra los esquemas DTD y XSD.
4. Si es válido, la información se muestra de forma estructurada y legible en el frontend.
5. El usuario puede entonces seleccionar un subconjunto de esta información.
6. Finalmente, el usuario descarga un nuevo XML que contiene únicamente la información seleccionada, listo para ser utilizado en otros procesos o plataformas.

Este aplicativo, por lo tanto, no solo sirve como una herramienta de prueba y demostración de las capacidades de validación y transformación del backend, sino que también sienta las bases para una utilidad que podría ayudar a los docentes a gestionar y adaptar su información curricular para diferentes fines.

# Fuentes de información

aplyca. (2022, septiembre 14). *NextJS: ¿el futuro de la web?¿por qué y cuándo usarlo?* Aplyca Tecnología SAS. https://www.aplyca.com/blog/nextjs-el-futuro-web-que-es-nextjs

Deyimar, A. (2020, febrero 4). Qué es React: Definición, características y funcionamiento. *Tutoriales Hostinger*. https://www.hostinger.com/mx/tutoriales/que-es-react

Diego Lázaro. (2018). *XML principios básicos*. https://diego.com.es/xml-principios-basicos

DigitalOcean. (2022, agosto 3). *Python XML to JSON, XML to Dict | DigitalOcean*. https://www.digitalocean.com/community/tutorials/python-xml-to-json-dict

*DTD y XML Schema.pdf*. (s/f). Recuperado el 30 de marzo de 2025, de https://ocw.uc3m.es/pluginfile.php/1582/mod\_page/content/18/DTD\_XML\_esquema.pdf

FastAPI. (2025). *FastAPI*. https://fastapi.tiangolo.com/

FasterCapital. (2024, junio 14). *Validación De Documentos Xml Con Dtd Y Xsd*. FasterCapital. https://fastercapital.com/keyword/validación-de-documentos-xml-con-dtd-y-xsd.html

Hernández, C. (2014, febrero 17). *Fundamentos XML | PPT*. https://www.slideshare.net/narutogore/fundamentos-xml

IBM. (2021, marzo 4). *Rational Software Architect 9.6.1*. https://www.ibm.com/docs/es/rational-soft-arch/9.6.1?topic=data-validating-xml

*Introduction to OWL.pdf*. (s/f). Recuperado el 30 de marzo de 2025, de https://kmi.open.ac.uk/events/iswc07-semantic-web-intro/pdf/3.%20Introduction%20to%20OWL.pdf

Lledó, G. (2010, enero 7). *Partes de un documento XML – Educación y nuevas tecnologías*. https://blogs.ua.es/gonzalo/2010/01/07/partes-de-un-documento-xml/

Maciek. (2024, marzo 21). XML Conversion Using Python in 2025. *Sonra*. https://sonra.io/xml-conversion-using-python/

Martínez, J. E. (2020, mayo 25). La web semántica (II): RDF, vocabularios, ontologías y consultas. *Adictos al trabajo*. https://adictosaltrabajo.com/2020/05/25/la-web-semantica-ii-rdf-vocabularios-ontologias-y-consultas/

Microsoft. (2024). *XML para no iniciados—Soporte técnico de Microsoft*. https://support.microsoft.com/es-es/office/xml-para-no-iniciados-a87d234d-4c2e-4409-9cbc-45e4eb857d44

Next.js. (2025). *Introduction | Next.js*. https://nextjs.org/docs

Python. (2025). *xml.etree.ElementTree—The ElementTree XML API*. Python Documentation. https://docs.python.org/3/library/xml.etree.elementtree.html

React. (2024). *React*. https://es.react.dev/

Sánchez, J. (2025). *Jorge Sánchez. Manual de XML. Validación de documentos XML*. https://jorgesanchez.net/manuales/xml/xml-validacion.html

ScrapingAnt. (2024, agosto 2). *How to Parse XML in Python | ScrapingAnt*. https://scrapingant.com/blog/python-parse-xml

Semantic Web Technologies. (2025). *Semantic Web Technologies: XML, RDF, OWL*. https://jmvidal.cse.sc.edu/talks/xmlrdfdaml/allslides.html

W3C. (2025). *OWL Web Ontology Language Reference*. https://www.w3.org/TR/owl-ref/

xml.etree.ElementTree. (2019). *xml.etree.ElementTree—The ElementTree XML API - Python 3.7.3 Documentation*. https://documentation.help/python-3-7-3/xml.etree.elementtree.html