

Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Introducción a la Programación y Computación 2
Escuela de Vacaciones - Junio 2025

Ing. Carlos Alberto Arias López
Tutora del Curso: Andrea Maria Cabrera Rosito



PROYECTO 2

OBJETIVO GENERAL

Desarrollar una solución integral que implemente una API que brinde servicios utilizando el protocolo HTTP bajo el concepto de programación orientada a objetos (POO).

OBJETIVOS ESPECÍFICOS

- Implementar un API a través de lenguaje Python que pueda ser consumida utilizando el protocolo HTTP.
- Utilizar el paradigma de programación orientada a objetos para construir software.
- Utilizar archivos XML como insumos para la comunicación con el API desarrollado.
- Utilizar expresiones regulares para extraer contenido de texto.

DESCRIPCIÓN GENERAL

Ha destacado en la empresa IPC2-Games con su anterior entrega, es por esto que le han ascendido y cambiado de área. Ahora, IPC2-GeneralManagement le ha solicitado un proyecto de desarrollo web orientado a la enseñanza. Su nuevo proyecto se llama **IPC2-AcadNet**.

IPC2-AcadNet será una plataforma web educativa que conectará a estudiantes con tutores especializados dentro del entorno académico. El objetivo principal es facilitar el seguimiento personalizado del aprendizaje, permitir la gestión eficiente de sesiones de tutoría, y brindar estadísticas útiles a docentes y administradores.

Es una plataforma de apoyo académico basada en una arquitectura web moderna:

- El frontend está desarrollado con Django, proporcionando interfaces limpias, seguras y administrables.
- El backend está potenciado por Flask, responsable del procesamiento lógico y analítico de los datos, incluyendo servicios inteligentes y estructuras optimizadas como matrices dispersas.

En este sistema, se hace uso de una matriz dispersa para manejar el progreso temático individualizado de los estudiantes, que se actualiza a medida que cada estudiante avanza en diferentes temas académicos ofrecidos por los tutores.

IMPLEMENTACIÓN

IPC2-AcadNet se transformará en una aplicación web basada en arquitectura cliente-servidor, diseñada para ser consumida como un servicio desde internet. El sistema permitirá convertir archivos XML que contienen información académica (como cursos, estudiantes y tutores) en un formato JSON estandarizado, almacenando temporalmente estos datos antes de su envío al servidor. Esto garantizará la consistencia de la información y modernizará la forma en que se gestionan los recursos educativos.

La solución incluirá una API desarrollada en Flask, encargada de manejar eficientemente las operaciones de transformación, validación y almacenamiento de datos educativos. Esta API también será responsable de administrar estructuras como matrices dispersas, que se utilizarán para representar el progreso temático de los estudiantes de forma eficiente.

El sistema se complementará con un frontend desarrollado en Django, que ofrecerá una interfaz intuitiva para estudiantes, tutores y administradores. Desde esta interfaz se podrá visualizar el avance académico, consultar temas, asignar actividades y generar reportes.

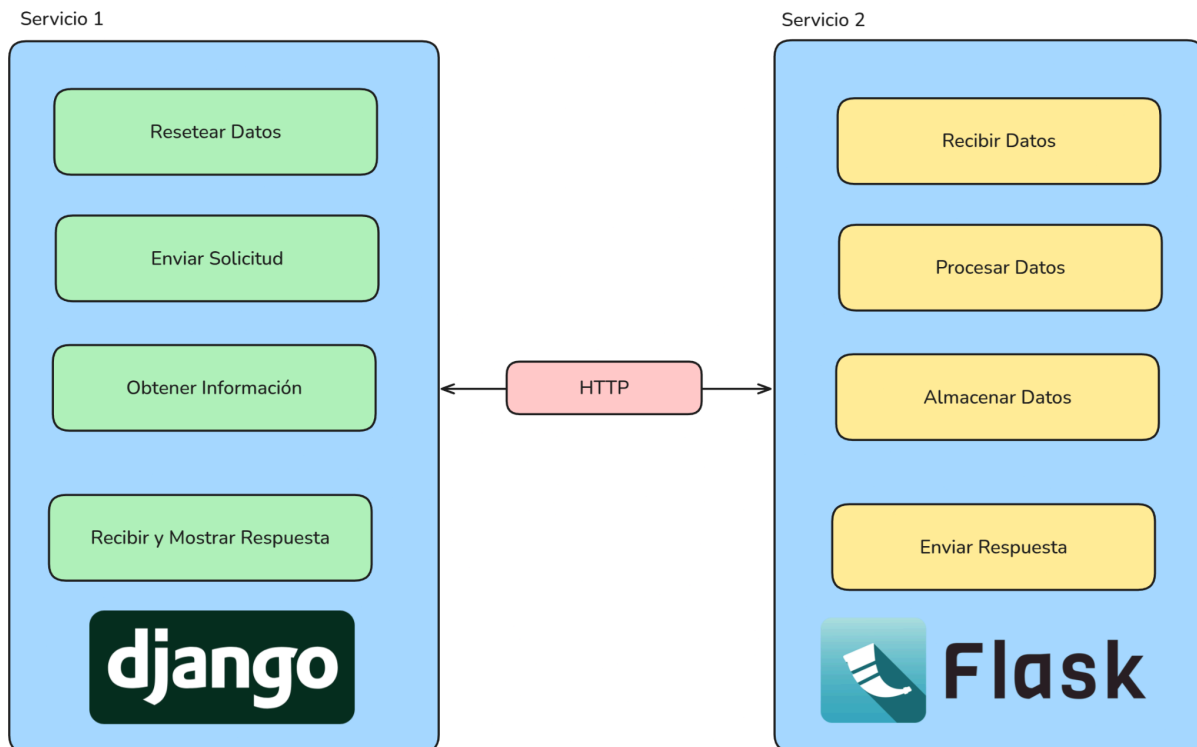
Esta evolución permitirá a IPC2-AcadNet adaptarse a las necesidades actuales de accesibilidad, personalización y escalabilidad en entornos educativos modernos, consolidándose como una herramienta integral para la gestión del aprendizaje académico.

ARQUITECTURA

La arquitectura cliente-servidor describe una relación entre un programa (cliente) que solicita un servicio o recurso a otro programa (el servidor). En el caso de IPC2-AcadNet, se ha solicitado el desarrollo de un software que pueda ser consumido desde internet como un servicio educativo interactivo.

Este software tendrá la capacidad de transformar archivos XML que contienen información académica —como cursos, estudiantes, temas y avance académico— hacia un formato JSON estandarizado. Este JSON será almacenado temporalmente antes de ser enviado al servidor, con el fin de validar y asegurar la consistencia de los datos.

Dado que se busca estandarizar y modernizar el sistema de almacenamiento, IPC2-AcadNet apunta a migrar las funciones académicas clave a un entorno virtual basado en la web, permitiendo así un acceso más dinámico, accesible y escalable desde cualquier lugar con conexión a internet.



Servicio 1 - Frontend

Este programa consiste en una aplicación Web y consistirá en un simulador de la aplicación principal, contendrá únicamente las funcionalidades necesarias para testear el buen funcionamiento de la API (Servicio 2), en esta aplicación se podrán mostrar los eventos que se procesarán y los datos estadísticos que fueron almacenados, al igual que las salidas en XML.

Para realizar el frontend deberá utilizarse el framework Django, el cual trabaja con el patrón MVT (Modelo-Vista-Template).

Acá, se tendrán como funcionalidades:

- Carga de Archivos XML.
- Inicio de Sesión.
- Consulta de Datos
- Reportes
- Etc.

Se describen estas más adelante.

Servicio 2 - Backend

Este servicio consiste en una API que brindará servicios utilizando el protocolo HTTP, su funcionalidad principal es procesar los datos recibidos del programa 1, luego de procesar los datos es necesario que estos sean almacenados en uno o varios archivos xml, algunos de estos archivos están especificados en la sección de archivos de entrada y salida, este servicio también tiene la funcionalidad de devolver los datos que fueron almacenados para que sean mostrados como se indica en la sección “Servicio 1 – Frontend / Componentes”.

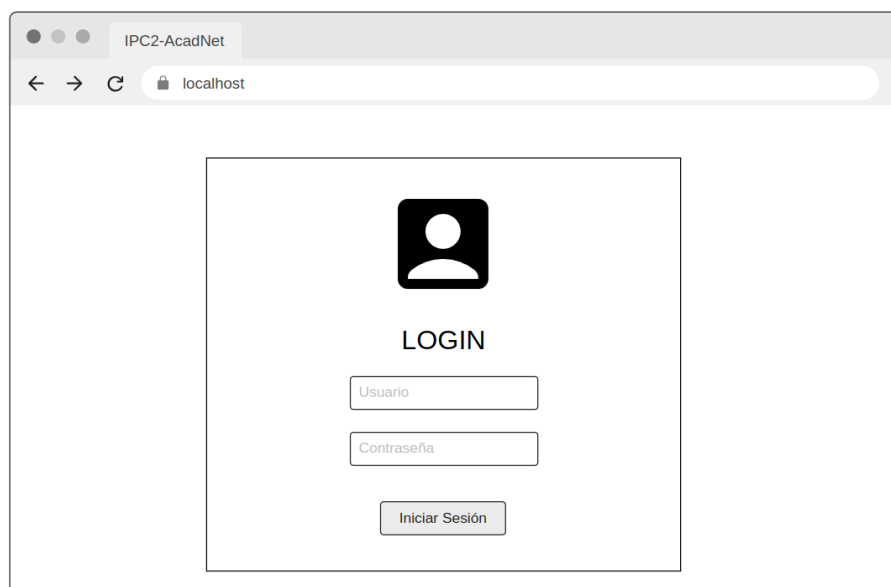
Para la realización de este servicio debe utilizarse el framework Flask. El estudiante deberá definir por su propia cuenta los métodos que necesitará para la realización de este servicio. Esto significa que debe implementar tantos métodos como necesite para consumir la API.

NOTA: *Durante la calificación de este proyecto, el Servicio 2 podrá ser consumido desde otro cliente, por ejemplo, Postman.*

Login

Esta es la pantalla principal del programa. Únicamente podrán iniciar sesión los usuarios identificados que se encuentren cargados en el programa. Existen 3 tipos de roles:

- Estudiantes
- Tutores
- Administrador



The image shows a web browser window with the title "IPC2-AcadNet". The address bar shows "localhost". The main content area displays a login form. At the top of the form is a black square icon with a white circle inside. Below the icon is the word "LOGIN" in bold. There are two input fields: the first is labeled "Usuario" and the second is labeled "Contraseña". Below these fields is a button labeled "Iniciar Sesión".

Nota: El usuario administrador deberá de existir siempre, debe de tener el usuario **AdminIPC2** y de contraseña **AdminIPC2771**. A continuación se describe por medio de los roles las funcionalidades que tendrá cada tipo de usuario.

Módulo de Administrador:

Solamente existirá un administrador, esta persona podrá:

CARGAR ARCHIVO DE ENTRADA:

Se desplegará una pantalla para gestionar la carga de los archivos de entrada con extensión .xml con los archivos de configuración. Se especifica en la sección de archivos de entrada y salida.

Antes de procesar los datos dentro del archivo de entrada, podrá editarlo dentro de la aplicación web o limpiar toda el área de edición de texto. Se le sugiere esta interfaz:

The screenshot shows a web browser window titled "IPC2-AcadNet" with the address bar set to "localhost". The interface is divided into a left sidebar and a main content area. The sidebar contains a user profile icon labeled "Admin" and three menu items: "Cargar Archivo", "Ver Usuarios", and "Información". The main content area is titled "CARGA DE ARCHIVOS" and features three buttons: "Cargar Archivo", "Procesar Datos", and "Limpiar Datos". Below these buttons are two text input fields, each containing the XML declaration: `<?xml version="1.0" encoding="UTF-8"?>` followed by four asterisks. The second section is titled "SALIDA" and contains a similar text input field.

Debe de aparecer después de procesar, un XML con la salida correspondiente que se explica más adelante.

VER USUARIOS:

Para revisar que se han configurado correctamente todos los usuarios, el administrador podrá ver estos desde la aplicación web.

The screenshot shows the same web browser window, but the main content area is now titled "REVISIÓN DE USUARIOS". The sidebar remains the same. Below the title, there is a table with three columns: "ID", "Usuario", and "Contraseña". The table has five rows, all of which are currently empty.

ID	Usuario	Contraseña

INFORMACIÓN:

En esta sección, se mostrarán sus datos de estudiante con el link en su repositorio a la carpeta de documentación.

The screenshot shows a web browser window with the title 'IPC2-AcadNet' and the address bar set to 'localhost'. The interface is divided into a left sidebar and a main content area. The sidebar contains a user profile icon, the name 'Admin', and three menu items: 'Cargar Archivo', 'Ver Usuarios', and 'Información'. The main content area is titled 'INFORMACIÓN' and contains a form with a large information icon. The form fields are: 'Nombre Completo:' followed by three dashed input boxes, 'Carnet:' followed by three dashed input boxes, and 'Link a la documentación:' followed by five dashed input boxes.

Módulo de Tutor:

Los tutores serán aquellos que podrán gestionar lo que son los horarios disponibles según los cursos cargados.

CONFIGURAR HORARIOS DE TUTORÍA:

Los tutores podrán gestionar sus horarios de junio por medio de un XML de forma masiva, dependiendo del curso que posean. El horario será cargado con el con el formato que se le definirá después.

XML

```
<?xml version="1.0"?>
<horarios>
  <curso codigo="XXX">
    Mi horario de curso es HorarioI: 09:40 HorarioF: 10:30
  </curso>
  <curso codigo="YYY">
    Elijo este horario HorarioI: 09:40 HorarioF: 10:30
  </curso>
  <curso codigo="ZZZ">
    HorarioI: 09:40 HorarioF: 10:30
  </curso>
</horarios>
```

De la cadena, deberá extraer HorarioI: **HH:mm** HorarioF: **HH:mm**.
Cualquier otro dato deberá de ser descartado. Si el código no coincide con algún curso que el tutor tenga asignado, no deberá de ser tomado en cuenta. Una vez configurados, el tutor podrá visualizar en una tabla sus horarios cargados.

HORARIOS

Cargar Archivo de Horarios

	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES
XXX - Curso1	10:40 - 11:30	10:40 - 11:30	10:40 - 11:30	10:40 - 11:30	10:40 - 11:30
YYY - Curso2	08:00 - 08:50	08:00 - 08:50	08:00 - 08:50	08:00 - 08:50	08:00 - 08:50
ZZZ - Curso3					
WWW - Curso4					

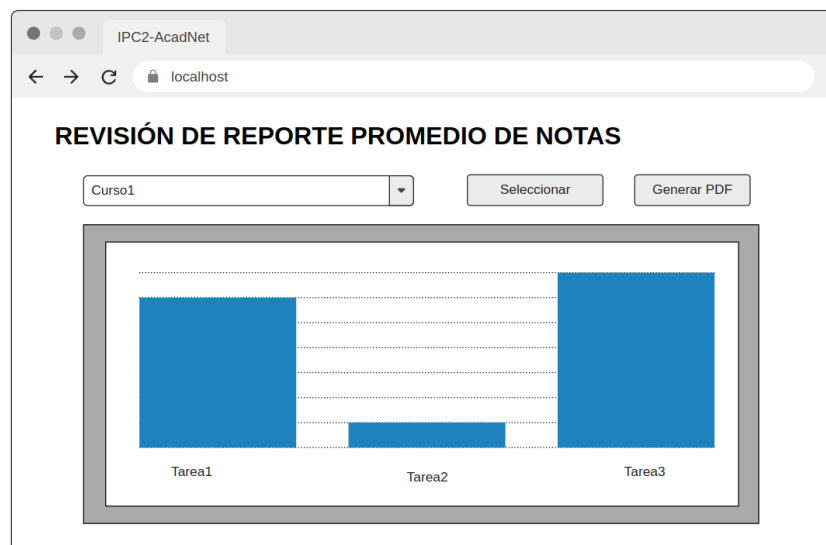
INGRESO NOTA:

Para el ingreso de notas, el tutor deberá de subir un archivo XML que describa el desarrollo que ha tenido un alumno en el curso del tutor. Es importante que esta información se guarde en una **matriz dispersa** creada por usted, es decir, utilizando POO, así podrá visualizar esta información después por medio de un reporte ordenado.

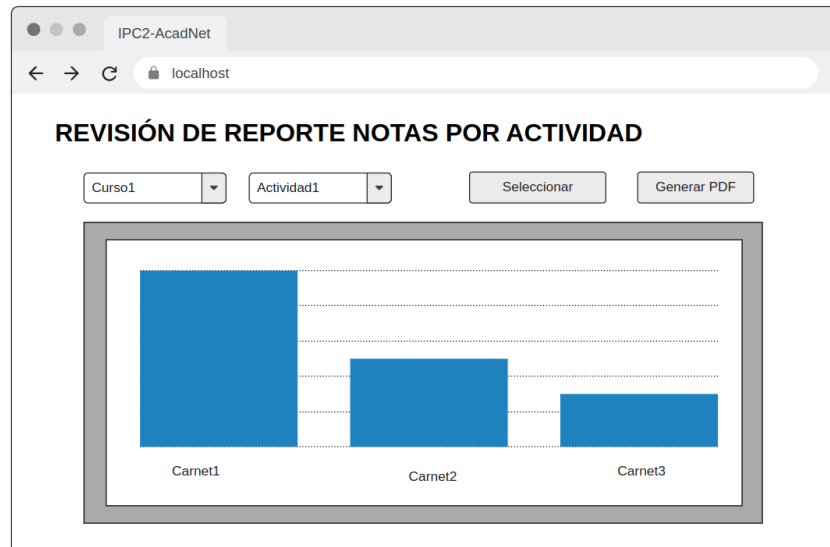
XML

```
<?xml version="1.0"?>
<curso codigo="XXXX">Nombre_del_curso</curso>
<notas>
  <actividad nombre="Tarea1" carnet="XXXX">90</actividad>
  <actividad nombre="Tarea2" carnet="XXX">60</actividad>
  <actividad nombre="Tarea3" carnet="YYY">40</actividad>
  <actividad nombre="Tarea4" carnet="ZZZ">100</actividad>
  <actividad nombre="Tarea5" carnet="XXX">10</actividad>
  ...
</notas>
```

En este caso, las filas serán las actividades que se indiquen en el XML y las columnas, el carnet de los alumnos. Ej:



- TOP de notas de cierta actividad para cierto curso: Se seleccionará un curso disponible para el tutor, después se seleccionará la actividad y se mostrará en un gráfico de manera ordenada de mayor a menor las notas.



Módulo de Estudiante:

Los estudiantes serán aquellos que revisen que la información cargada a sus cursos esté correcta.

REVISIÓN NOTAS:

Un estudiante al ingresar, podrá revisar las notas si es que su tutor ya se las ha cargado. Dicha información estará disponible en una tabla.

The screenshot shows a web application interface titled "MIS NOTAS". At the top, there is a dropdown menu labeled "Curso1" and a button labeled "Consultar". Below these controls is a table with two columns: "Tarea" and "Nota". The table contains four rows of data:

Tarea	Nota
Tarea1	100
Tarea2	70
Tarea3	100
Tarea4	100

Archivo de Entrada:

Las configuraciones iniciales, se cargarán por medio de un archivo XML con la siguiente información:

```
XML
<?xml version="1.0"?>
<configuraciones>
  < cursos>
    <curso codigo="770">Nombre_del_curso</curso>
    <curso codigo="771">Nombre_del_curso</curso>
    <curso codigo="772">Nombre_del_curso</curso>
    ...
  </cursos>
  <tutores>
    <tutor registro_personal="1111" contrasenia="1234">nombre del
tutor</tutor>
    <tutor registro_personal="2222" contrasenia="1234">nombre del
tutor</tutor>
    <tutor registro_personal="3333" contrasenia="1234">nombre del
tutor</tutor>
    ...
  </tutores>
  <estudiantes>
    <estudiante carnet="1234" contrasenia="1234">nombre del
estudiante</estudiante>
    <estudiante carnet="5678" contrasenia="1234">nombre del
estudiante</estudiante>
    <estudiante carnet="1122" contrasenia="1234">nombre del
estudiante</estudiante>
    ...
  </estudiantes>
  <asignaciones>
    <c_tutores>
      <tutor_curso codigo="770">1111</tutor_curso>
      <tutor_curso codigo="771">1111</tutor_curso>
      <tutor_curso codigo="772">2222</tutor_curso>
      ...
    </c_tutores>
    <c_estudiante>
      <estudiante_curso codigo="770">1234</estudiante_curso>
      <estudiante_curso codigo="770">1234</estudiante_curso>
      <estudiante_curso codigo="771">1122</estudiante_curso>
      ...
    </c_estudiante>
  </asignaciones>
</configuraciones>
```

Donde en asignaciones, el atributo código describe el código del curso que debe de haber sido cargado con anterioridad. Si no existe, la asignación no se llevará a cabo. Luego, dentro de las asignaciones, los valores serán ya sea el registro de personal del tutor o del estudiante si fuese el caso.

Archivo de Salida:

La salida después de procesarse el archivo de entrada tendrá que tener este formato:

```
XML
<?xml version="1.0"?>
<configuraciones_aplicadas>
  <tutores_cargados>#Total</tutores_cargados>
  <estudiantes_cargados>#Total</estudiantes_cargados>
  <asignaciones>
    <tutores>
      <total>#total</total>
      <correcto>#asignaciones_correctas</correcto>
      <incorrecto>#asignaciones_incorrectas</incorrecto>
    </tutores>
    <estudiantes>
      <total>#total</total>
      <correcto>#asignaciones_correctas</correcto>
      <incorrecto>#asignaciones_incorrectas</incorrecto>
    </estudiantes>
  </asignaciones>
</configuraciones_aplicadas>
```

Donde se muestra el total de tutores cargados correctamente, número total de estudiantes cargados correctamente. Igualmente, en asignaciones, se tendrá el total de asignaciones de cada tipo (estudiantes, tutores), cuántas de estas fueron correctas e incorrectas.

Consideraciones:

Debe utilizarse versionamiento para el desarrollo del proyecto. Se utilizará la plataforma Github en la cual se debe crear un repositorio en el que se gestionará el proyecto. **Se debe realizar dos releases.** En este proyecto **SI** está permitido el uso de estructuras propias de Python (list, dict, tuple, set), exceptuando la matriz dispersa que deberá de realizar con POO. Para el desarrollo de gráficas en el frontend deberá utilizar la librería chartjs o plotly.

Se deberá agregar a su respectivo auxiliar como colaborador del repositorio. El último release será el release final y se deberá de realizar antes de entregar el proyecto en la fecha estipulada.

Documentación

Para que el proyecto sea calificado, el estudiante deberá entregar la documentación utilizando el formato de ensayo definido para el curso. En el caso del proyecto, el ensayo debe tener entre 2 y 5 páginas de contenido, este máximo no incluye los apéndices o anexos donde se pueden mostrar modelos y diseños utilizados para construir la solución. Es obligatorio incluir el diagrama de clases que modela la solución de software presentada por el estudiante.

RESTRICCIONES

- El nombre del repositorio debe de ser **IPC2_VJunio_Proyecto2_Carnet**. Debe de unir a la auxiliar como colaboradora para tener derecho a calificación: **AndreaCabrera01**.
- El estudiante debe entregar la documentación solicitada para poder optar a la calificación. (Ensayo y Diagrama de clases).
- Los archivos de entrada no podrán modificarse.
- Los archivos de salida deben llevar la estructura mostrada en el enunciado obligatoriamente.
- Deben existir 2 releases uno por cada semana, de esta manera se corrobora el avance continuo del proyecto. Se definirá una penalización por cada release faltante.
- Se calificará el último release almacenado en el repositorio Github. Los cambios realizados después de ese release no se tomarán en cuenta.
- Cualquier caso de copia parcial o total tendrá una nota de 0 y será reportada a la Escuela de Ciencias y Sistemas.
- En este proyecto **está permitido el uso de estructuras propias de Python** (list, dict, tuple, set, etc). **Solamente deberá de desarrollar la solución con POO sobre la matriz dispersa.**
- El estudiante debe subir la documentación solicitada al repositorio con el formato definido, si no se entrega con el formato requerido no se calificará.
- **NO HABRÁ PRÓRROGA.**
- **Solo se permitirá la utilización de las librerías y frameworks indicados en este enunciado.**

ENTREGA

- La entrega será el 26 de Junio a más tardar las 23:59.
- La entrega es por medio de UEDi, si no hay entrega del link del repositorio, no se calificará.
- La documentación debe de estar subida en el repositorio y en el release a calificar.
- La calificación será **presencial**, donde se indicará la hora y lugar unos días antes.