Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Introducción a la Programación y Computación 2 - N
Ing. Jaime Francisco Yuman Ramirez
Aux. Marcos Sebastián Velasquez Cabrera

Proyecto

Objetivos

- Aplicar los conocimientos de programación para construir un sistema funcional.
- Aprender los conceptos de frontend y backend.
- Desarrollar un sistema interactivo entre el usuario y la información mediante dos aplicaciones utilizando el concepto cliente servidor.
- Desarrollar una API en Python, que servirá como servidor en donde se ejecutarán todas las tareas que el sistema debe realizar.
- Desarrollar un sitio web mediante un framework que permita trabajar con HTML y CSS, para la manipulación y presentación de información.
- Aplicar conocimientos sobre el manejo de versiones en Git.
- Conocer el uso de ficheros para la lectura, consulta y escritura de información en archivos XML.
- Generar reportes mediante el uso de la herramienta Graphviz.
- Presentar una solución tecnológica en una reunión virtual, donde exponga cómo resolverá el problema de estudio..

Descripción

La empresa Desktop Records, decide que es momento de presentar su información de una manera más cómoda para los usuarios, por lo que se desea implementar un sitio web capaz de procesar, manipular y presentar la información de la empresa.

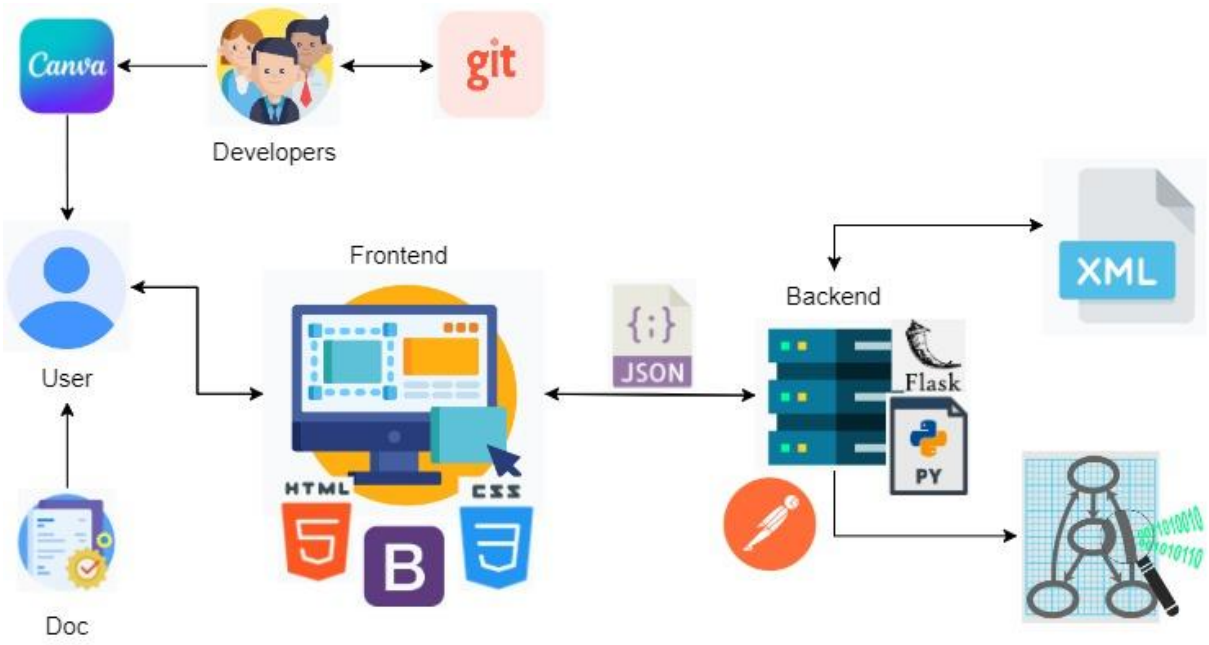
El sistema estará compuesto por dos aplicaciones principales, un servidor de Python que funcionará como Backend, y una plataforma Web que servirá como Frontend para presentar la información.

Dado que actualmente el mercado de desarrollo de software tiene bastante competencia, usted y su equipo de desarrollo deberán de realizar una presentación en donde se pretenda convencer a los directivos de Desktop Records de optar por su solución tecnológica sobre las demás.

Finalmente, se debe de desarrollar documentación técnica que permita al usuario final conocer qué es y cómo funciona el producto.

Arquitectura

La arquitectura que se debe implementar es la siguiente, en donde se distinguen cómo se debe comunicar cada componente del proyecto.



Especificaciones del programa

Actualmente, la empresa maneja los datos de empleados, discos musicales y regiones. Cada unos de estos datos es manejado en archivos XML individuales. La aplicación deberá poder soportar la carga de estos archivos para poder manipular los datos.

Se solicita persistencia de datos, por lo cual, todas las operaciones realizadas, se deberán de poder ver en los archivos físicos generados.

empleados.xml

```xml
<?xml version="1.0" encoding = "utf-8" ?>
<empresa>
    <departamento departamento="Contabilidad">
        <empleado id="1">
            <nombre>José Ernesto</nombre>
            <puesto>Gerente general</puesto>
            <salario>4000.00</salario>
        </empleado>
        <empleado id="2">
            <nombre>Daniel Pérez</nombre>
            <puesto>Contador</puesto>
            <salario>2000.00</salario>
        </empleado>
    </departamento>
    <departamento departamento="RRHH">
        <empleado id="3">
            <nombre>Marta Torres</nombre>
            <puesto>Reclutadora</puesto>
            <salario>3000.00</salario>
        </empleado>
    </departamento>
</empresa>
```

discos.xml

D: > Proyectos Python > py archivos xml > discos.xml

```xml
<?xml version="1.0" encoding = "utf-8" ?>
<catalog>
    <cd>
        <title>empire burlesque</title>
        <artist>bob dylan</artist>
        <country>usa</country>
        <company>columbia</company>
        <price>10.90</price>
        <year>1985</year>
    </cd>
    <cd>
        <title>greatest hits</title>
        <artist>dolly parton</artist>
        <country>usa</country>
        <company>rca</company>
        <price>9.90</price>
        <year>1982</year>
    </cd>
    <cd>
        <title>&amp;quot;empire burlesque&amp;quot;</title>
        <artist>bee gees</artist>
        <country>uk</country>
        <company>polydor</company>
        <price>10.90</price>
        <year>1998</year>
    </cd>
</catalog>
```

Esta vez, se requiere también, que se lleve el control de los países en donde la plataforma tiene alcance. Por lo cual es necesario procesar un archivo XML extra, el cual se llamará mundo.XML.

De los países se necesita manejar información de:

- continente
- nombre del país
- moneda
- capital
- idioma
- registro de población según el año

La sintaxis del nuevo archivo es el siguiente:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<mundo>
  <continente name="Europa">
    <pais moneda="Euro">
      <nombre>Monaco</nombre>
      <capital>Ciudad de Monaco</capital>
      <idioma>Frances</idioma>
      <poblacion year="2019" unit="thousands">38.964</poblacion>
    </pais>
    <pais moneda="Euro">
      <nombre>Austria</nombre>
      <capital>Viena</capital>
      <idioma>Aleman</idioma>
      <poblacion year="2019" unit="millions">8.859</poblacion>
    </pais>
  </continente>
  <continente name="America">
    <pais moneda="Quetzal">
      <nombre>Guatemala</nombre>
      <capital>Ciudad de Guatemala</capital>
      <idioma>Español</idioma>
      <poblacion year="2019" unit="millions">16.6</poblacion>
    </pais>
  </continente>
</mundo>
```

Backend

Se debe implementar un servidor con Python y Flask para declarar las peticiones GET y POST que posteriormente serán llamadas desde la aplicación de Frontend. En el servidor es donde se harán todas las operaciones de lectura y escritura de datos de los archivos XML. Queda a discreción del estudiante el puerto a utilizar para levantar el servidor.

Todas las respuestas de las peticiones deben de enviarse en formato JSON. De la misma manera, cada petición realizada al servidor, debe hacerse con formato JSON.

A continuación se describen las peticiones que debe contener el servidor:

GET - /empleados

Devolverá la información de todos los empleados de cada departamento.

POST - /empleadoNombre

Se enviará un nombre y la petición devolverá la información de los empleados que contengan el nombre indicado.

POST - /empleadoDepartamento

Se enviará un departamento y la petición devolverá la información de los empleados que pertenezcan al departamento indicado.

POST - /empleadoSueldo

Se enviará un sueldo y la petición devolverá la información de los empleados que contengan el sueldo indicado.

POST - /agregarEmpleado

Se enviará una estructura JSON con la información del empleado (departamento, ID, nombre, puesto, salario) y la petición ejecutará código para poder ingresar el empleado al departamento. No se evaluarán los casos de errores ID repetido, o departamento inexistente, por lo que no es necesario realizar esas validaciones.

POST - /modificarEmpleado

Se enviará una estructura JSON con la información del empleado (ID, nombre, puesto, salario) y la petición ejecutará código para poder modificar el empleado con el ID enviado.

POST - /eliminarEmpleado

Se enviará una estructura JSON con la información del empleado (ID) y la petición ejecutará código para poder eliminar el empleado del departamento.

GET - /reporteEmpleados

Esta petición ejecutará código para generar el gráfico de Graphviz con la información de los empleados.

GET - /discos

Devolverá la información de todos los discos.

POST - /discoTitulo

Se enviará un nombre y la petición devolverá la información de los discos que contengan el nombre indicado.

POST - /discoYear

Se enviará un año y la petición devolverá la información de todos los discos que pertenezcan al año indicado.

POST - /discoArtista

Se enviará un nombre y la petición devolverá la información de todos los discos que contengan el nombre indicado.

POST - /agregarDisco

Se enviará una estructura JSON con la información del disco (título, artista, país, compañía, precio y año) y la petición ejecutará código para poder ingresar el disco al catálogo.

POST - /modificarDisco

Se enviará una estructura JSON con la información del disco (título, artista, país, compañía, precio y año) y la petición ejecutará código para poder modificar el disco con el título indicado.

POST - /eliminarDisco

Se enviará una estructura JSON con la información del disco (título) y la petición ejecutará código para poder eliminar el disco del departamento.

GET - /reporteDiscos

Esta petición ejecutará código para generar el gráfico de Graphviz con la información de los discos del catálogo.

GET - /paises

Devolverá la información de todos los países de cada continente.

POST - /paisMoneda

Se enviará un tipo de moneda y la petición devolverá la información de los países que contengan la moneda indicada..

POST - /paisIdioma

Se enviará un idioma y la petición devolverá la información de todos los países que contengan el idioma indicado.

POST - /contienente

Se enviará un nombre de continente y la petición devolverá la información de todos los países de ese continente.

POST - /agregarPais

Se enviará una estructura JSON con la información del país (continente, nombre, moneda, capital, idioma, población) y la petición ejecutará código para poder ingresarlo.

POST - /modificarPais

Se enviará una estructura JSON con la información del país (continente, nombre, moneda, capital, idioma, población) y la petición ejecutará código para poder modificarlo.

GET - /reporteRegiones

Esta petición ejecutará código para generar el gráfico de Graphviz con la información de los países registrados en la plataforma.

El equipo de desarrollo puede agregar más peticiones si lo considera oportuno.

Frontend

Se debe implementar una plataforma web en donde el usuario pueda interactuar con la información. El frontend realizará las peticiones GET y POST del backend para poder ejecutar las funcionalidades de la aplicación.

Queda a discreción del equipo de desarrollo el framework o librería a utilizar.

La forma en la cual se presentan las vistas de la aplicación al usuario, queda a discreción del equipo de desarrollo, ya que se evaluará la creatividad y el diseño que decian darle a su aplicación. A continuación se muestran las funcionalidades del sistema:

Ver empleados

Mostrará el listado de todos los empleados de cada departamento en el sistema. Es necesario usar la petición GET /empleados.

Buscar según nombre

Mostrará el listado de todos los empleados que tengan el nombre indicado. Es necesario usar la petición POST /empleadoNombre

Buscar según departamento

Mostrará el listado de todos los empleados del departamento indicado. Es necesario usar la petición POST /empleadoDepartamento.

Buscar según sueldo

Mostrará el listado de todos los empleados de cada departamento con el sueldo indicado. Es necesario usar la petición POST /empleadoSueldo

Agregar empleado

Se ingresan los datos del nuevo empleado y se enviará la información en formato JSON para poder usar la petición POST /agregarEmpleado.

Modificar empleado

Se ingresan los datos del empleado y se enviará la información en formato JSON para poder usar la petición POST /modificarEmpleado

Eliminar empleado

Se ingresa el ID del empleado y se enviará la información en formato JSON para poder usar la petición POST /eliminarEmpleado

Reporte empleados

Se generará la imagen del reporte de empleados. Se debe utilizar la petición GET /reporteEmpleados. Se debe de mostrar la imagen en el frontend.

Ver discos

Mostrará el listado de todos los discos. Es necesario usar la petición GET /discos.

Buscar discos según nombre

Mostrará el listado de todos los discos que tengan el nombre indicado. Es necesario usar la petición POST /discoTitulo

Buscar discos según año

Mostrará el listado de todos los discos del año indicado. Es necesario usar la petición POST /discoYear.

Buscar discos según artista

Mostrará el listado de todos los discos del artista indicado. Es necesario usar la petición POST /discoArtista

Agregar disco

Se ingresan los datos del nuevo disco y se enviará la información en formato JSON para poder usar la petición POST /agregarDisco

Modificar disco

Se ingresan los datos del disco y se enviará la información en formato JSON para poder usar la petición POST /modificarDisco

Eliminar disco

Se ingresa el título del disco y se enviará la información en formato JSON para poder usar la petición POST /eliminarDisco

Reporte discos

Se generará la imagen del reporte de empleados. Se debe utilizar la petición GET /reporteDiscos. Se debe de mostrar la imagen en el frontend.

Ver regiones

Mostrará el listado de todos los países. Es necesario usar la petición GET /países.

Buscar países según moneda

Mostrará el listado de todos los países con la moneda indicada. Es necesario usar la petición POST /paisMoneda

Buscar países según idioma

Mostrará el listado de todos los países con el idioma indicado. Es necesario usar la petición POST /paisIdioma.

Buscar países según continente

Mostrará el listado de todos los países del continente indicado. Es necesario usar la petición POST /continente

Agregar país

Se ingresan los datos del nuevo país y se enviará la información en formato JSON para poder usar la petición POST /agregarPais

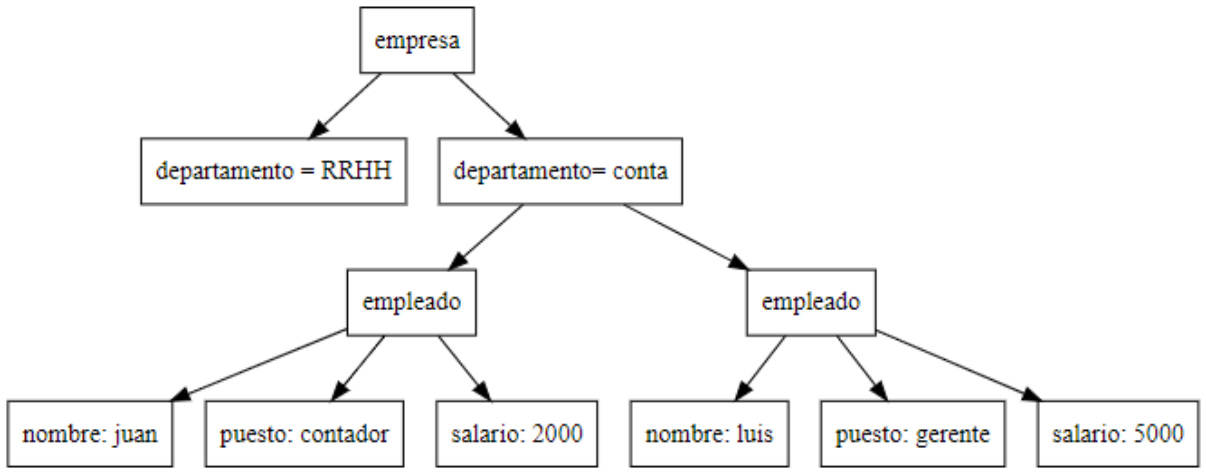
Reporte regiones

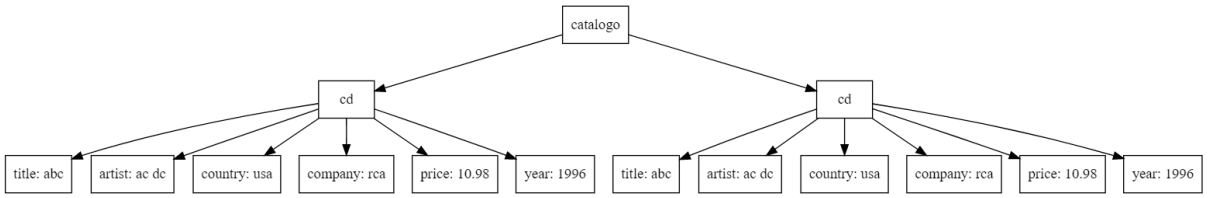
Se generará la imagen del reporte de los países de cada continente. Se debe utilizar la petición GET /reporteRegiones. Se debe de mostrar la imagen en el frontend.

Queda a discreción del equipo de desarrollo, la manera en la cuál se presentará el listado de resultados de cada funcionalidad. Y respecto a los reportes de graphviz, se debe de poder visualizar la imagen en el frontend.

Reportes

Para ambos reportes, se deben generar gráficos utilizando Graphviz, con todos los nodos y nodos hijos, que permita ver la estructura completa del archivo. Ejemplos:





Documentación

Se deben de crear los manuales técnicos y de usuario. Ya que es un proyecto profesional, ambos manuales deben de tener el formato **IEEE**, con introducción, objetivos del manual y contenido del manual.

Manual técnico

Debe de contener la información sobre todas las peticiones GET y POST utilizadas en el servidor del backend. También debe de contener la descripción de todas las herramientas utilizadas para el desarrollo de las aplicaciones (VS Code, Framework Angular, descripción de Flask, Github, etc).

Manual de usuario

Debe de ser una guía para el usuario, por lo cuál debe de contener información sobre el aspecto y funcionalidades del Frontend. Incluir capturas de pantalla de la aplicación para poder facilitar el entendimiento del lector.

Formato IEEE - Título en español y en ingles

Presentación del proyecto

Se debe crear una presentación con el objetivo de vender su producto a la empresa musical. En la reunión, se debe presentar al equipo de trabajo, presentar el producto y mostrar las ventajas que ofrece para el consumidor. Se recomienda usar Canva, ya que con el correo institucional se tiene acceso a la versión Pro de esta herramienta, pero puede ser realizada en cualquier plataforma o herramienta.

Observaciones

- La aplicación de Frontend debe de contar con un diseño intuitivo al usuario, es decir, que contenga un diseño que sea entendible y capaz de alojar usuarios nuevos y tengan noción de cómo se utiliza el sistema.
- La calificación será vía Google Meet asignado al laboratorio y el equipo de desarrollo debe encender su cámara al inicio de la misma.
- La calificación tendrá una duración máxima de 20 minutos, por lo cuál se debe de tomar en cuenta la duración de su presentación.
- Durante la calificación se utilizará la herramienta POSTMAN para asegurar que se utilizaron peticiones GET y POST para el desarrollo del proyecto.
- Durante la calificación se estarán realizando preguntas sobre el desarrollo de la aplicación para comprobar que fue realizado por el estudiante
- De encontrar copias se tendrá una nota de 0 y se reportará a las autoridades según corresponda.

Restricciones

- El proyecto se debe de realizar en grupo de **máximo 4 integrantes**.
- El servidor del backend deberá ser desarrollado en el lenguaje de programación Python utilizando Flask.
- El framework o librería para el desarrollo de la aplicación del frontend queda a discreción del grupo.
- El editor de texto para desarrollar la aplicación queda a discreción del estudiante.
- Se deberán utilizar los lenguajes y herramientas indicadas, de lo contrario no tendrán derecho a calificación. Por ejemplo, Flask para el servidor del backend, o formato JSON para las peticiones GET y POST.
- Se realizarán preguntas acerca de la implementación del código y de la lógica utilizada para resolver el problema.

Repositorio de Github

Se debe de crear un repositorio privado para el proyecto con el nombre **IPC2_Proyecto_G#** (Ejemplo: IPC2_Proyecto_G4). El mismo debe de contener las siguientes carpetas:

- **Frontend:** Código fuente de la aplicación para el frontend.
- **Backend:** Código fuente del servidor para el backend.
- **Documentación:** Manual técnico y manual de usuario, ambos en formato PDF.
- **Presentación:** Presentación en formato PDF para la exposición del proyecto.

Entregables

- Repositorio de GitHub **Privado** con el código fuente.

FECHA DE ENTREGA: 30 DE JUNIO DE 2022 A LAS 23:59 en UEDI.