

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Introducción a la Programación y Computación 2
Escuela de Vacaciones Junio 2024



Inga. Claudia Liceth Rojas Morales
Tutor de curso: Rodrigo Alejandro Hernández de León

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 una 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

Debido al éxito generado a través de la aplicación de escritorio de IPCmarket se ha deseado que este deje de ser manejada a manera local si no que pueda ser consumida por medio de una página web para con ello expandir su alcance y poder mostrar variedad de estadísticas con relación tanto de los productos, los usuarios como de las compras a través de gráficas entendibles y dinámicas.

IMPLEMENTACIÓN

La arquitectura cliente – servidor habla acerca de una relación entre un programa (cliente) solicita un servicio o recurso de otro programa (el servidor).

Se ha solicitado que construya un software que pueda ser consumido desde internet como un servicio. Dicho software será capaz de transformar los archivos XML actuales hacia un formato JSON, el JSON debe de ser almacenado previo al envío hacia el servidor ya que al pretender la estandarización del almacenaje se espera migrar la aplicación de app hacia un eCommerce.

ARQUITECTURA

La arquitectura cliente – servidor habla acerca de una relación entre un programa (cliente) solicita un servicio o recurso de otro programa (el servidor).

Se ha solicitado que construya un software que pueda ser consumido desde internet como un servicio. Dicho software será capaz de transformar los archivos XML actuales hacia un formato JSON, el JSON debe de ser almacenado previo al envío hacia el servidor ya que al pretender la estandarización del almacenaje se espera migrar la aplicación de eCommerce hacia un entorno virtual.

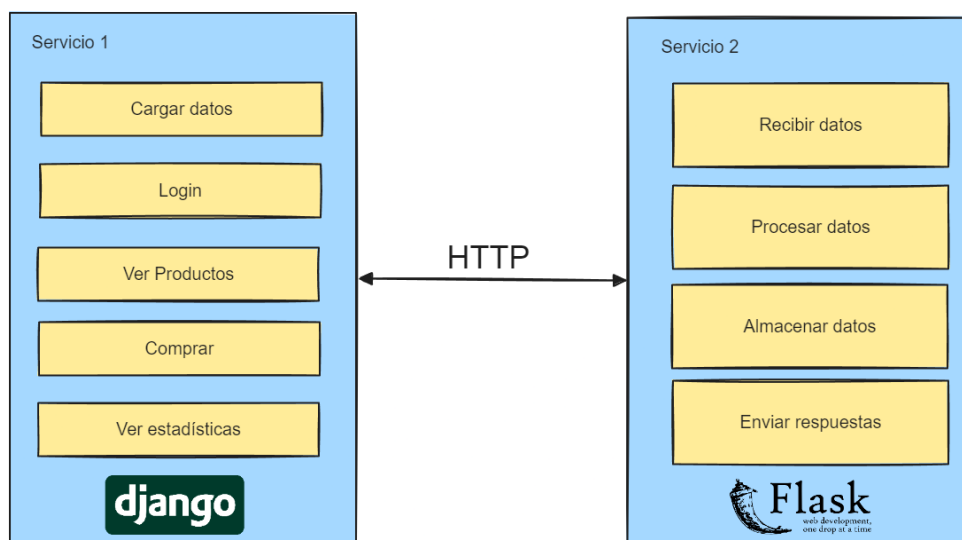


Figura 1: Arquitectura de la aplicación

SERVICIO FRONTEND

Consiste en una aplicación web, que contará con las funcionalidades tales como:

- Lectura de archivos XML.
- Inicio de sesión de usuarios.
- Comprar productos.
- Ver estadísticas de compra.
- Agregar productos al carrito.

Se espera que en la aplicación se pueda visualizar los archivos XML previo al envío, así como el XML correspondiente con los datos que el servidor retorna.

Dado que es una aplicación donde se realicen compras, es necesario contar con dos tipos de

usuario:

- Administrador
- Comprador

Para el manejo de la carga de información se le solicitara a usted que únicamente acepte la carga de archivos XML, procese la información por medio de estructuras de datos y expresiones regulares y almacene la información por medio de XML.

En el caso de los usuarios tienen que ingresar sus credenciales y estos usuarios deben de estar previamente registrados en el sistema por medio del administrador.

Para acceder como administrador (va a ser el único usuario quemado en el sistema ósea que quiere decir que no estará en la lista de usuarios) va a contar con las siguientes credenciales:

- Usuario: AdminIPC2
- Contraseña: IPC2VJ2024

LOGIN

En este apartado únicamente inician sesión los usuarios compradores o el administrador y tiene que ser la primera vista en la cual el usuario mire cuando abra la aplicación.

A hand-drawn sketch of a web browser window. The browser's address bar is empty. The page content features the title "IPCMarket Online" in a handwritten font. Below the title are two input fields: the first is labeled "ID:" and the second is labeled "Contraseña". Both fields are empty. Below these fields is a rectangular button labeled "Login".

MODULO ADMINISTRADOR

En este módulo únicamente ingresa el administrador y contará con las siguientes funcionalidades:

CARGA MASIVA

La aplicación contará con una opción que permitirá que el administrador pueda cargar archivos de tipo XML donde se pueden cargar de 1 a muchos archivos XML y no borrarán el contenido que tengan en las listas al cargar un nuevo archivo XML. Dentro de los archivos XML que se cargarán serán los siguientes:

- Usuarios:

```
<?xml version="1.0" encoding="UTF-8"?>
<compradores>
  <usuario id="1" password="12345">
    <nombre>Rodrigo Hernández</nombre>
    <edad>23</edad>
    <email>rodri@gmail.com</email>
    <telefono>12345678</telefono>
  </usuario>
  <usuario id="2" password="12345">
    <nombre>Alejandro de León</nombre>
    <edad>25</edad>
    <email>ale@gmail.com</email>
    <telefono>87654321</telefono>
  </usuario>
</compradores>
```

Los usuarios se guardarán en un **XML** que estará detallado más adelante y será la persistencia de la aplicación y cada usuario contará con los datos de:

- Id: Que puede ser un entero o puede contener letras, pero va a ser único en toda la aplicación y valide que no se repita el mismo id en un usuario ya que con este id ingresa a la aplicación al iniciar sesión.
- Nombre: Nombre del comprador.
- Edad: Edad del comprador.
- Email: Es el correo del comprador y tiene que validar con una expresión regular que si sea la sintaxis correcta del correo electrónico.
- Teléfono: Es el número del comprador, tiene que validar que sean enteros y que obligatoriamente debe de contener 8 dígitos.

- Productos:

```
<?xml version="1.0" encoding="UTF-8"?>
<productos>
  <producto id="IPC1-1">
    <nombre>Computadora de Escritorio</nombre>
    <precio>5,000.00</precio>
    <descripcion>Computadora de escritorio con procesador Intel Core i7, 16 GB de RAM, 1 TB de almacenamiento.</descripcion>
    <categoria>Tecnología</categoria>
    <cantidad>10</cantidad>
    <imagen>C:\imagenesP1IPC2\imagen1.jpg</imagen>
  </producto>
  <producto id="IPC1-2">
    <nombre>Frijoles</nombre>
    <precio>2.50</precio>
    <descripcion>1 libra de frijoles negros</descripcion>
    <categoria>Comida</categoria>
    <cantidad>50</cantidad>
    <imagen>C:\imagenesP1IPC2\imagen2.jpg</imagen>
  </producto>
</productos>
```

Los productos se guardarán en un **XML** que estará detallado más adelante y será la persistencia de la aplicación y cada producto contará con los datos de:

- Id: Es el identificador único de cada producto.
- Precio: Es el precio del producto, tiene que contener decimales.
- Descripción: Es la descripción del producto.
- Categoría: Es la categoría del producto.
- Cantidad: Únicamente tiene que ser entero y es la cantidad disponible de productos en el sistema.
- Imagen: Es la ruta de la imagen del producto (puede ser un link de internet).
- Empleados de IPCMarket:

```
<?xml version="1.0" encoding="UTF-8"?>
<empleados>
  <empleado codigo="1">
    <nombre>Rodrigo Hernández</nombre>
    <puesto>Vendedor</puesto>
  </empleado>
  <empleado codigo="2">
    <nombre>Alejandro de León</nombre>
    <puesto>Vendedor</puesto>
  </empleado>
</empleados>
```

Los empleados se guardarán en un **XML** que estará detallado más adelante y será la persistencia de la aplicación y cada empleado contará con los datos de:

- Código: Es el identificador único de cada empleado.
- Nombre: Es el nombre completo del empleado.
- Puesto: Es el puesto que desempeña el empleado.
- Actividades de IPCMarket:

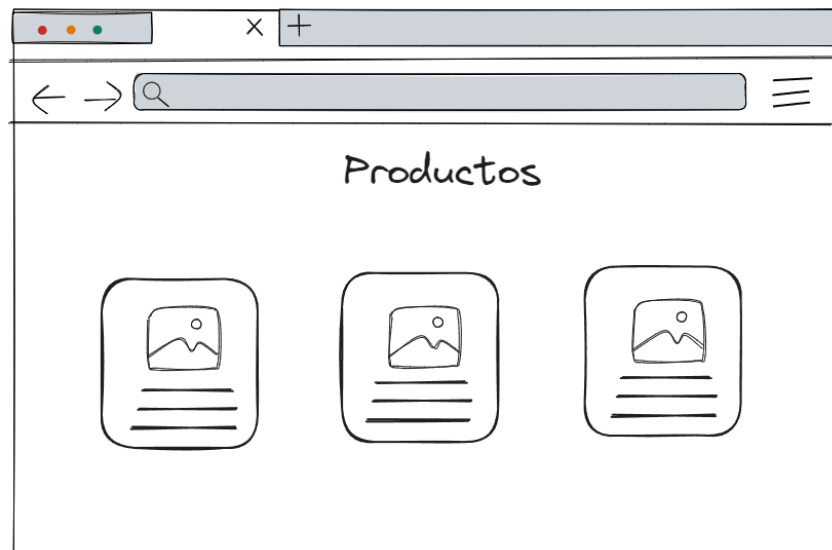
```
<?xml version="1.0" encoding="UTF-8"?>
<actividades>
  <actividad id="1">
    <nombre>Atención al Cliente</nombre>
    <descripcion>Atender a los clientes de la tienda</descripcion>
    <empleado>1</empleado>
    <dia hora="14">2</dia>
  </actividad>
  <actividad id="2">
    <nombre>Limpieza de estanterías</nombre>
    <descripcion>Limpiar y ordenar las estanterías del pasillo 1</descripcion>
    <empleado>1</empleado>
    <dia hora="7">1</dia>
  </actividad>
</actividades>
```

Las actividades se guardarán en un **XML** que estará detallado más adelante y será la persistencia de la aplicación y cada actividad contará con los datos de:

- Id: Es el identificador único de cada actividad.
- Nombre: Es el nombre de la actividad.
- Descripción: Es el detalle de la actividad.
- Empleado: Es el id del empleado.
- Dia: Es el número de día (únicamente se representa del 1 al 7 que significa de lunes a Domingo).
- Hora: Es la hora en que se realizará la actividad (Estará en formato de 24h ósea únicamente aparecerán números del 0 al 23)

VER PRODUCTOS

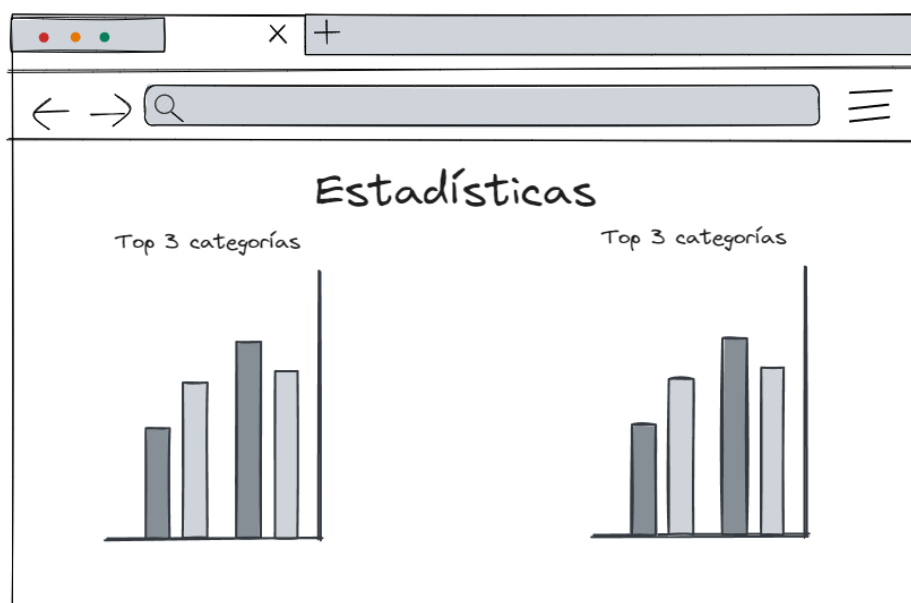
El administrador puede ver todos los productos cargados en el sistema.



ESTADÍSTICAS

El administrador puede permitir al administrador las siguientes estadísticas en graficas de barras:

- Top 3 de categorías con más productos.
- Top 3 de productos con más cantidad disponible.



REPORTES

Se le solicitará que se logren mostrar los siguientes reportes:

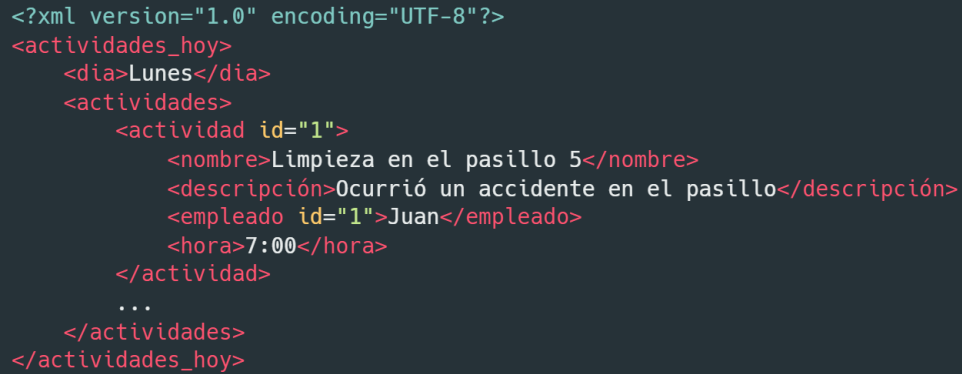
1. Compras: Se mostrará un XML de todas las compras que se realizaron en el sistema.

```
<?xml version="1.0" encoding="UTF-8"?>
<compras>
  <compra numero="1">
    <usuario id="1">Rodrigo Hernández</usuario>
    <Total>5000.00</Total>
    <productos>
      <producto id="1">
        <nombre>Computadora de Escritorio</nombre>
        <cantidad>1</cantidad>
      </producto>
      <producto id="2">
        <nombre>Laptop</nombre>
        <cantidad>1</cantidad>
      </producto>
    </productos>
  </compra>
  <compra numero="2">
    <usuario id="1">Rodrigo Hernández</usuario>
    <Total>10.00</Total>
    <productos>
      <producto id="3">
        <nombre>Frijol</nombre>
        <cantidad>2</cantidad>
      </producto>
    </productos>
  </compra>
</compras>
```

Este XML debe contar con lo siguiente:

- Compras: Cuenta con todas las compras realizadas en el sistema.
- Compra: Contará con un ID único y detalla la compra.
- Usuario: Contará con el ID del usuario que realiza la compra junto con su nombre.
- Total: Es la suma total de la compra.
- Productos: Es el listado de productos que compró el usuario.
- Producto: Contará con el id del producto que se compró.
- Nombre: Es el nombre del producto.
- Cantidad: Es la cantidad del producto que compró el usuario.

2. Actividades_hoy: Se mostrarán todas las actividades del día actual (por ejemplo si el día de hoy es Lunes, mostrará las actividades del día Lunes) junto con los detalles de su empleado como lo muestra la siguiente imagen:



```
<?xml version="1.0" encoding="UTF-8"?>
<actividades_hoy>
  <dia>Lunes</dia>
  <actividades>
    <actividad id="1">
      <nombre>Limpieza en el pasillo 5</nombre>
      <descripción>Ocurrió un accidente en el pasillo</descripción>
      <empleado id="1">Juan</empleado>
      <hora>7:00</hora>
    </actividad>
    ...
  </actividades>
</actividades_hoy>
```

Este XML debe de contener lo siguiente:

- Actividades_hoy: Contendrá toda la información de las actividades del día actual.
- Día: Mostrará el nombre del día de la semana (si en la entrada era 1, mostrará Lunes y así sucesivamente).
- Actividades: Contará con todas las actividades del día actual.
- Actividad: Contará con el id de la actividad junto con sus otros atributos.
- Nombre: Es el nombre de la actividad.
- Descripción: Es la descripción de la actividad.
- Empleado: Cuenta con el ID y nombre del empleado.
- Hora: Muestra la hora de la actividad en formato "HH:00"

AYUDA

Se desplegará 2 opciones, una para visualizar información de los estudiantes y otra para visualizar la documentación del programa.

MODULO DE USUARIO

En este módulo únicamente ingresan los usuarios que iniciaron sesión en la aplicación y contendrá lo siguiente:

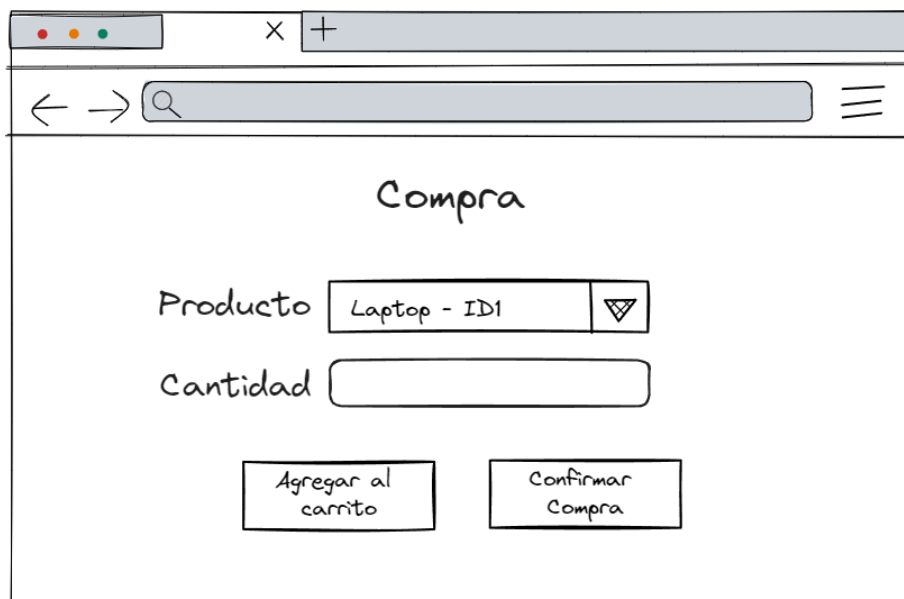
PRODUCTOS

Los usuarios pueden ver todos los productos cargados en el sistema.



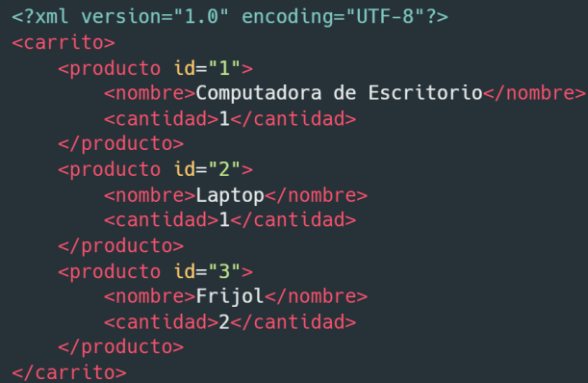
COMPRA

Los usuarios pueden seleccionar un producto y lo puede agregar a un Carrito de compras donde el usuario puede adjuntar la cantidad del producto que seleccionó siempre y cuando no supere la cantidad máxima que contiene del producto en el sistema, al completar el carrito de compras el usuario puede seleccionar la opción de confirmar compra para que se registre su compra y se vacíe el carrito.



VER CARRITO

Los usuarios pueden ver su carrito de compras donde se mostrará como un XML que contendrá el siguiente formato (puede verlo en la página o exportarlo):



```
<?xml version="1.0" encoding="UTF-8"?>
<carrito>
  <producto id="1">
    <nombre>Computadora de Escritorio</nombre>
    <cantidad>1</cantidad>
  </producto>
  <producto id="2">
    <nombre>Laptop</nombre>
    <cantidad>1</cantidad>
  </producto>
  <producto id="3">
    <nombre>Frijol</nombre>
    <cantidad>2</cantidad>
  </producto>
</carrito>
```

SERVICIO BACKEND

Este servicio consiste en una API que brindara servicios utilizando el protocolo HTTP, su funcionalidad principal es procesar los datos recibidos del servicio de frontend, luego de procesar los datos es necesario que estos sean almacenados en un archivo XML, este servicio también tiene la funcionalidad de devolver los datos que fueron almacenados para que sean mostrados como anteriormente se indica.

Dentro de los endpoints que se espera como mínimo que realice son:

- **/cargausuarios:** Entra el XML de usuarios para procesarlos.
- **/cargaproductos:** Entra el XML de productos para procesarlos.
- **/cargaempleados:** Entra el XML de productos para procesarlos.
- **/cargaactividades:** Entra el XML de productos para procesarlos.
- **/login:** Para el inicio de sesión de los usuarios.
- **/añadircarrito:** Para añadir el producto al carrito.
- **/comprar:** Para comprar los productos del carrito.

Y los demás endpoints que considere necesarios.

PERSISTENCIA

La aplicación contendrá una capa de persistencia donde la información cargada será guardada en un XML general de persistencia, queda a discreción del estudiante el formato en que almacenará su información en el XML.

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

CONSIDERACIONES

Debe utilizarse versionamiento para el desarrollo del proyecto. Se utilizará la plataforma Github en la cual se debe usar el mismo repositorio en donde realizó su primer proyecto. Se debe realizar un reléase y con el nombre de la versión del reléase que en este caso sería (v2.0.0) para este proyecto y se deberá de realizar antes de entregar el proyecto en la fecha estipulada. Además, cada estudiante debe trabajar una rama la cual tendrá de nombre su *no_carnet*. Y estarán uniendo sus avances en la rama main, para comprobar la colaboración de los avances del proyecto.

DOCUMENTACIÓN

Para que el proyecto sea calificado, el grupo de estudiantes deberá entregar la documentación utilizando el formato de ensayo definido para el curso. En el caso del proyecto, el ensayo puede tener un mínimo de 4 y un máximo de 7 páginas de contenido, este máximo no incluye los apéndices o anexos donde se pueden mostrar modelos y diseños utilizados para construir la solución. Este informe debe expresar con claridad el diseño de objetos ideado para resolver este proyecto por lo que debe expresar el diagrama de clases, diagrama de casos de uso y los diagramas de actividades de los algoritmos más importantes.

Debe de tomar en cuenta que es un ensayo formal, por lo que se calificará tanto la redacción, como ortografía y presentación.

RESTRICCIONES

- Solo se permitirá la utilización de los IDEs discutidos en el laboratorio.
- Uso obligatorio de programación orientada a objetos (POO).
- El nombre del repositorio debe de ser IPC2_ProyectoVJ2024_#Grupo, con la carpeta **Proyecto2**.
- Los estudiantes deben de entregar la documentación solicitada para poder optar a la calificación.
- Los archivos de entrada no podrán modificarse.
- Se calificará el release realizado previo a la fecha de entrega. No se calificará dado que se dé el caso que existan modificaciones de código en fechas posteriores a la entrega.
- Para dudas concernientes al proyecto se utilizarán los foros en UEDI de manera que todos los estudiantes puedan ver las preguntas y las posteriores respuestas.
- No es permitido el uso de JavaScript para el envío ni para la recepción de datos, únicamente es permitido para las estadísticas donde se usen gráficas (chartjs).
- Para el backend debe utilizarse el framework Flask mientras que para el frontend debe utilizarse Django, **NO ESTA PERMITIDO EL USO DE LIBRERIAS PARA DISEÑO DE INTERFACES GRAFICAS DE ESCRITORIO.**
- **COPIAS TOTALES O PARCIALES SERÁN REPORTADOS A LA ESCUELA Y OBTENDRÁN NOTA DE 0 PUNTOS.**
- **NO HABRÁ PRÓRROGA.**

ENTREGA

- La entrega será el día **miércoles 26 de junio** antes de las 23:59.
- La entrega será por medio de la UEDI.