Desarrollo web en Radiodiagnóstico

Sergio Díaz Prados[1], Antonio Jesús Morillo Gil[1], Francisco Javier Navarro Sánchez [1]

[1]Hospital Universitario Virgen de las Nieves, Granada

Objetivos docentes:

Objetivo principal: Desarrollar una página web de divulgación Radiológica

1. ¿Por qué aprender programación?
2. Proyecto en local: Como mostrar una Rx al mundo
3. Despliegue en Internet
4. Imagen médica en recursos didácticos digitales
5. Resultados

¿Por qué aprender programación web?

Si quisiéramos desarrollar una página web podríamos pensar en adquirir una existente en las múltiples plataformas que las ofrecen… ***¿Por qué hacerla nosotros desde 0? ¡Por muchas razones!:***

Personalización completa: Control **TOTAL** sobre la estructura y el diseño SIN LIMITARNOS A PLANTILLAS , permitiendo crear de forma personalizada interfaces para usuarios o desarrollo de herramientas de medición de imágenes, integraciones con sistemas de PACS…

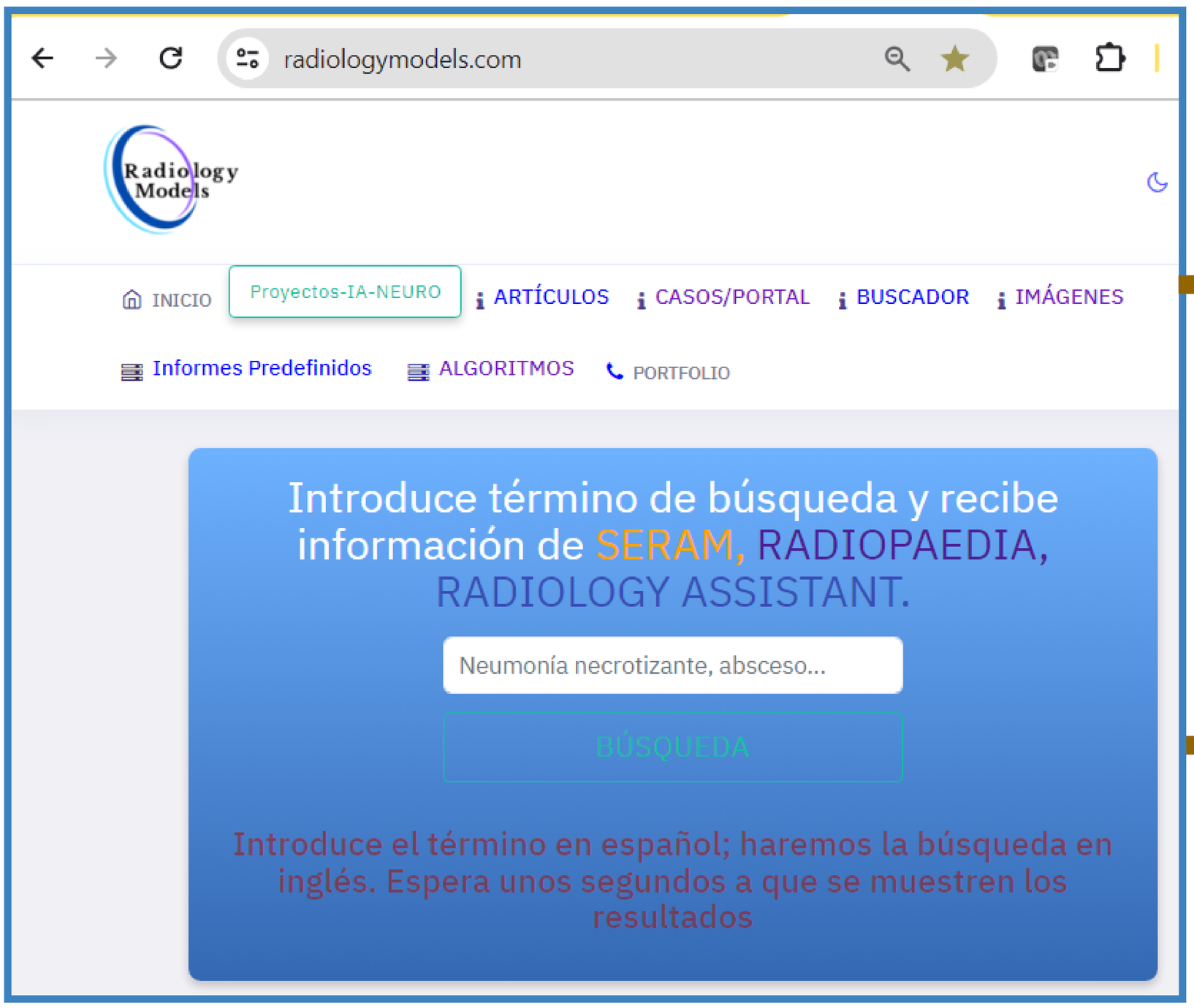
Rendimiento: Óptimo, al evitar el exceso de funcionalidad que se produce en las plataformas tipo blogging

Seguridad: Herramientas integradas para proteger tu sitio web contra amenazas

Escalabilidad, flexibilidad en la base de datos, cifrado de datos (permitiendo cumplir con la regulación médica) …

Proyecto en local

Front-end y Back-end: En una página web podemos distinguir dos aspectos fundamentales. El apartado estético «lo que ve el usuario» y la arquitectura de la página «lo que hay detrás de la pantalla». En cada uno de estos aspectos se emplean unos lenguajes de programación u otros



Back-end: *Lo que hace cada pestaña*

Front-end: *diseño de la página*

Analogía aclaratoria*: El programador dedicado a Front-end sería el pintor de una casa mientras que el dedicado a Back-end sería el obrero encargado de hacer los cimientos y su interior*

Proyecto en local

¿Por qué utilizar Python para desarrollo back-end?

1. ***Facilidad de Aprendizaje:*** Sintaxis sencilla que reduce la curva de aprendizaje para novicios

2. Frameworks Potentes: como Django y Flask que simplifican el desarrollo web y aceleran el proceso de desarrollo

3. Versatilidad: Python se utiliza también en inteligencia artificial, haciéndolo útil para una variedad de proyectos

4. Otros: Escalabilidad, Seguridad, Integración de Terceros (APIs), bibliotecas…

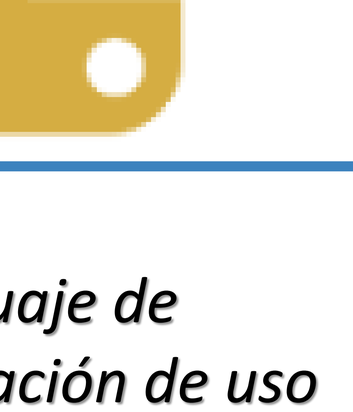
Leguaje de programación de uso abierto

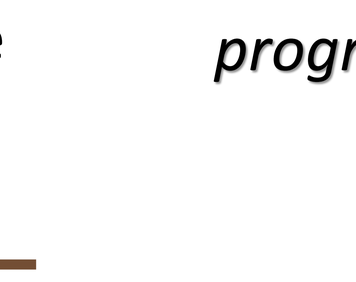
Lenguajes de programación front-end

1. ***HTML:*** Estructura el contenido que se muestra y su posición en la pantalla

2. CSS: Aporta las diferencias de color y diseño a cada página web

3. JS: Permite la interacción y los efectos visuales



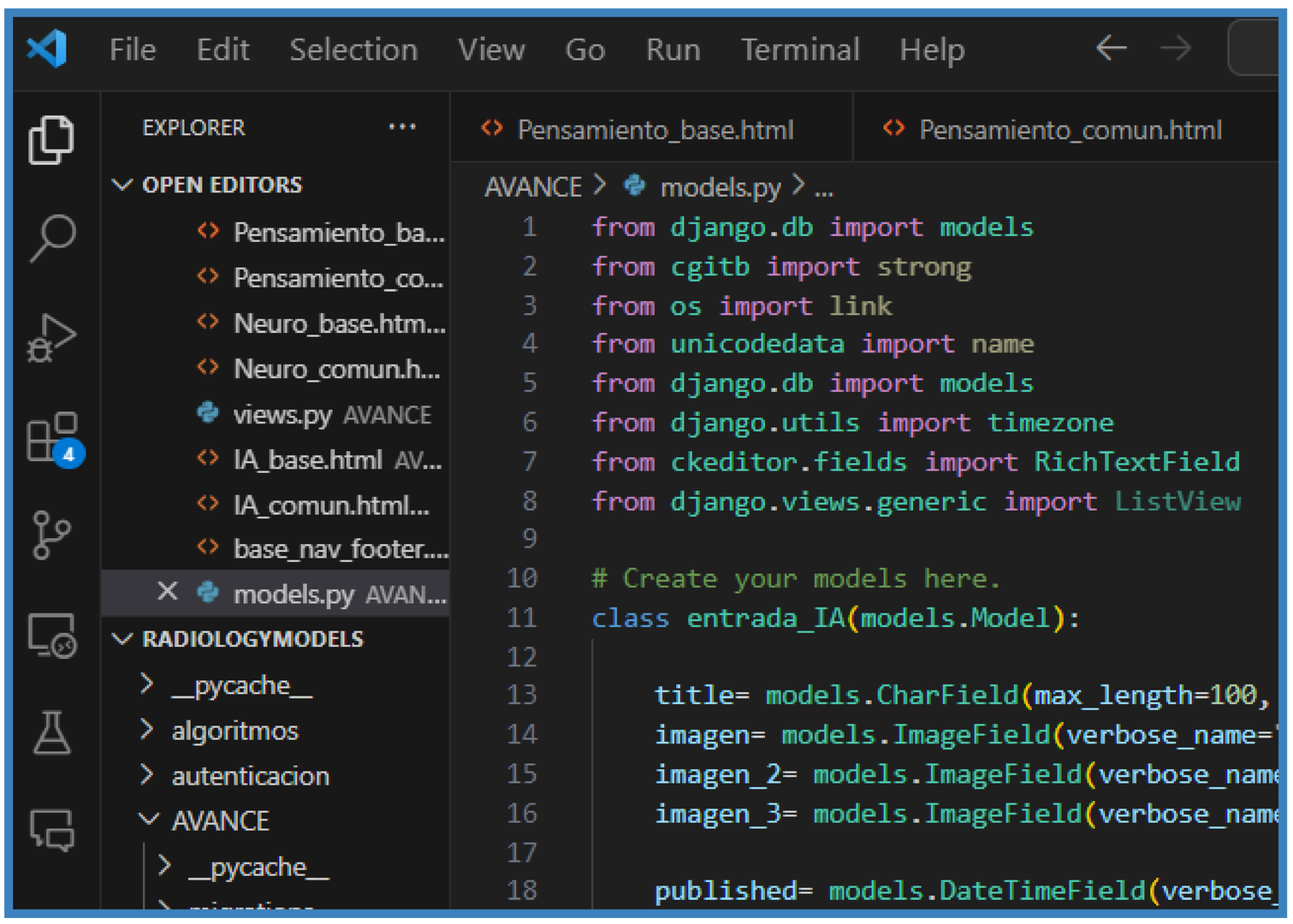


Leguajes de uso abierto

Proyecto en local

Lo primero… instalación de un editor de código: Nos proporciona herramientas específicas para la escritura y corrección del código, resaltando la sintaxis y permitiendo la lectura y comprensión del código. También permite la gestión organizada en carpetas y la integración con Git para facilitar la gestión, intercambio y guardado del proyecto

Existen muchos editores, pero el más utilizado es Visual Studio Code



En este proyecto se ha utilizado la programación orientado a objetos:

Se basa en la idea de crear clases que representan objetos del mundo real o conceptos. Una clase define las propiedades y comportamientos (atributos y métodos) que los objetos tendrán

Ejemplo de visor de editor de código

A continuación, vamos a ver como podemos desarrollar nuestro proyecto en una versión local, es decir, en nuestra computadora, antes de subirla a Internet

Proyecto en local

FUNCIONAMIENTO MVC (Modelo-Vista-Controlador) llamado en Django "MTV" (Modelo- Templatetag -Vista). El enrutamiento de URL conecta las solicitudes del usuario con las vistas adecuadas y con la base de datos para recupera datos según sea necesario:

1. Modelo (Model):

Representa la estructura de datos de tu aplicación, su almacenamiento y recuperación. En Django, los modelos se crean como «clases» y se encarga de la interacción con la base de datos

```python
class tcabdomen(models.Model):

    title= models.CharField(max_length=100, verbose_name="Título")
    subtitulo= models.CharField(max_length=400, verbose_name="Subtítulo", default= "Subtítulo")

    published= models.DateTimeField(verbose_name="Fecha de publicación", default=timezone.now)

    imagen1= models.ImageField(verbose_name="Imagen 1", upload_to="TC", null=True, blank= True)
    datosimagen1= models.CharField(max_length=500, verbose_name="Contenido imagen 1",default= "1")
    imagen2= models.ImageField(verbose_name="Imagen 2 ", upload_to="TC", null=True, blank= True)
    datosimagen2= models.CharField(max_length=500, verbose_name="Contenido imagen 2",default= "2")
    imagen3= models.ImageField(verbose_name="Imagen 3", upload_to="TC", null=True, blank= True)
    datosimagen3= models.CharField(max_length=500, verbose_name="Contenido imagen 3",default= "3")
    imagen4= models.ImageField(verbose_name="Imagen 4", upload_to="TC", null=True, blank= True)
    datosimagen4= models.CharField(max_length=500, verbose_name="Contenido imagen 4",default= "4")
    video = models.FileField(upload_to="TC", blank=True,null=True)
    video2 = models.FileField(upload_to="TC", blank=True,null=True)
```

Ejemplo de clase de nuestro proyecto

2. Plantillas HTML (Template):

Como se presentan los datos. Permiten la incrustación dinámica de los mismos con etiquetas y marcadores especiales

```
<main class="container">
  <div class="p-4 p-md-5 mb-4 text-white  bg-dark">
    <div class="col-md-6 px-0">
      <h1 class="display-4 fst-italic">{{posts.title}}</h1>
      <p class="lead my-3">{{posts.subtitulo}}</p>
      </div>
  </div>
  <div class="row mb-2">
    <div class="col-md-6">
      <div class="row g-0 border  overflow-hidden flex-md-row mb-4 shadow-sm h-md-250 position-relative">
        <div class="col p-4 d-flex flex-column position-static">
          <h4 class="d-inline-block mb-2 text-primary">{{posts.autor}}</h4>
          <h4 class="mb-0" > {{posts.explicaautor}}</h4>
        </div>
        <div class="col-auto d-none d-lg-block">
          {% if posts.image %}
            <img src="{{posts.imagenautor.url}}" alt="Promoción"   width="180" height="160">
            {% endif %}
```

Ejemplo de plantilla asociada de nuestro proyecto

Proyecto en local

FUNCIONAMIENTO MVC (Modelo-Vista-Controlador) llamado en Django "MTV" (Modelo- Templatetag -Vista). El enrutamiento de URL conecta las solicitudes del usuario con las vistas adecuadas y con la base de datos recuperando datos según sea necesario. Este enfoque MVC/MTV facilita la organización y el desarrollo de aplicaciones web robustas y escalables:

3. Vista (View):

Define cómo se procesan las solicitudes del usuario, recuperando los datos del modelo y renderizando la respuesta que se envía al navegador del usuario. En Django, las vistas se definen como funciones o clases Python y se conectan a URLs específicas

```python
def blog_abdomentc (request,post_id):
    posts= tcabdomen.objects.get(id=post_id)
    n= tcabdomen.objects.all()
    current_url= request.resolver_match.view_name

    return render (request, "entrada_tc.html", {'posts':posts, 'cate':n, 'base_url': current_url, 'titulo': "TC ABDOMEN" })
```

Ejemplo de vista de nuestro proyecto

4. URL Routing (Enrutamiento de URL):

Define cómo se relacionan las URLs ingresadas por el usuario con las vistas correspondientes. En Django, configuras las rutas URL en un archivo llamado `*urls.py*` en tu aplicación, donde especificas qué vista debe manejar cada URL

```python
from django.urls import path
from . import views

urlpatterns = [
    path('tc/patologia', views.patologiatc),
    path('tc/patologia/abdomen/<int:post_id>/', views.blog_abdomentc, name="tcabdomen"),
```

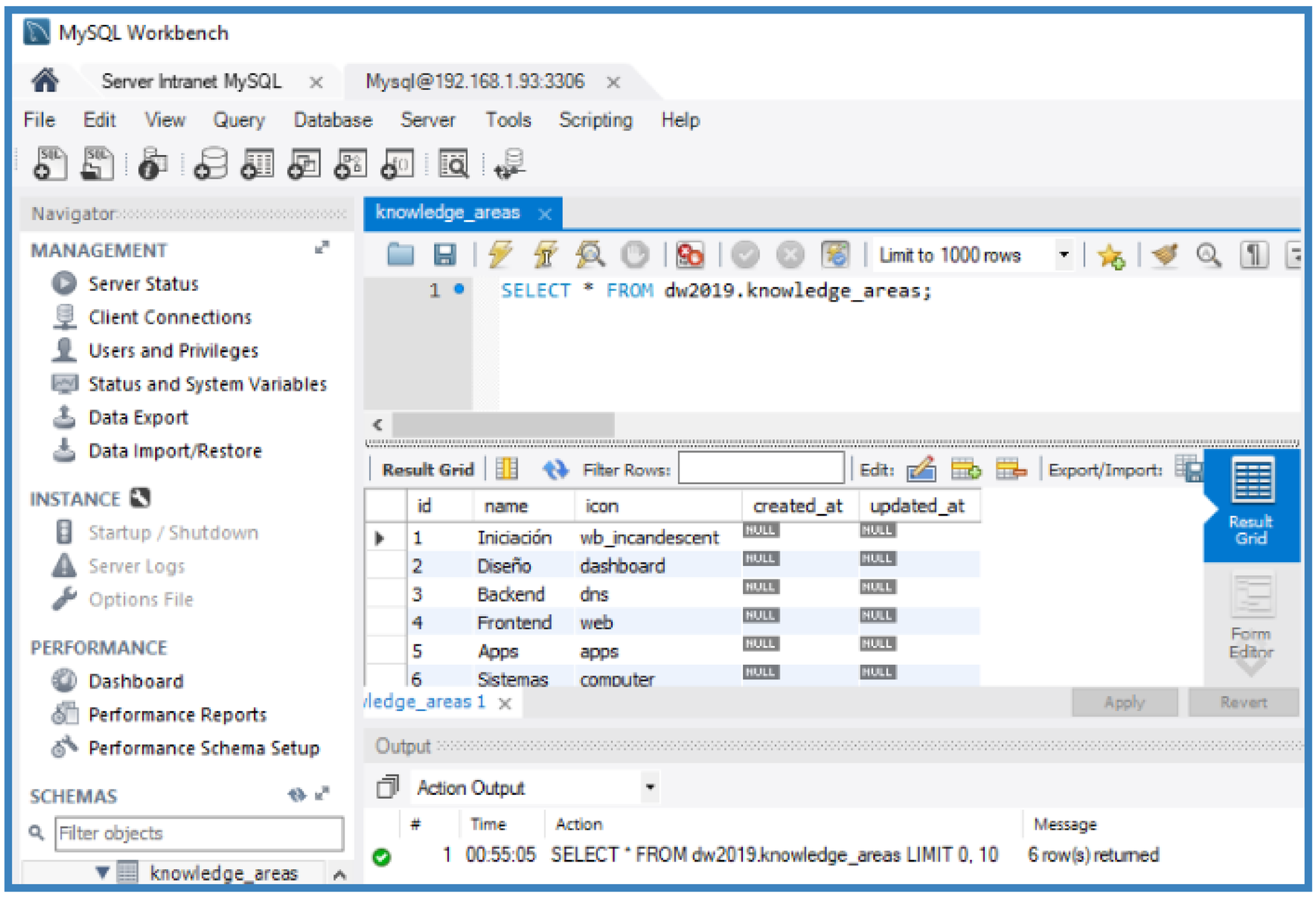
Ejemplo de URL asociada a la vista correspondiente

Proyecto en local

FUNCIONAMIENTO MVC (Modelo-Vista-Controlador) llamado en Django "MTV" (Modelo- Templatetag -Vista). El enrutamiento de URL conecta las solicitudes del usuario con las vistas adecuadas y con la base de datos recuperando datos según sea necesario. Este enfoque MVC/MTV facilita la organización y el desarrollo de aplicaciones web robustas y escalables:

5. Base de Datos:

Configuramos la base de datos en el archivo *`settings.py`*, donde puedes elegir entre varias bases de datos como SQLite, PostgreSQL, MySQL. Django se encarga de interactuar con la base de datos según los modelos definidos



Ejemplo de como se guardan los datos en una base de datos relacional

6. Controlador (Control):

Las vistas pueden realizar operaciones como crear, leer, actualizar y eliminar registros en la base de datos, y luego renderizar las plantillas con los datos recuperados. En Django, el controlador está implícito en las vistas. Las vistas son el punto de entrada para procesar las solicitudes y manejar la lógica del proyecto

Proyecto en local

Ejemplo de las diapositivas anteriores explicando donde encontramos todos los elementos comentados anteriormente: URL, modelo, vista y plantilla

- ***Círculo negro:*** URL

- ***Círculos rojos:*** Atributos de nuestro modelo. Todas las entradas de abdomen seguirán este formato, ya que todas hemos hecho que dependan del modelo *TCabdomen* que incluye los campos: título, subtítulo, autor, imágenes, principales hallazgos

- ***Estrellas:*** El diseño de nuestra página web viene determinado por la plantilla HTML indicando, por ejemplo, los colores de los títulos y la disposición de la información

La vista: La vista «llama» a este modelo (información) cuando el usuario quiere visitar «abscesos hepáticos»

Proyecto en local

FUNCIONAMIENTO MVC (Modelo-Vista-Controlador) llamado en Django "MTV" (Modelo- Templatetag -Vista). El enrutamiento de URL conecta las solicitudes del usuario con las vistas adecuadas y con la base de datos, recuperando datos según sea necesario. Este enfoque MVC/MTV facilita la organización y el desarrollo de aplicaciones web robustas y escalables:

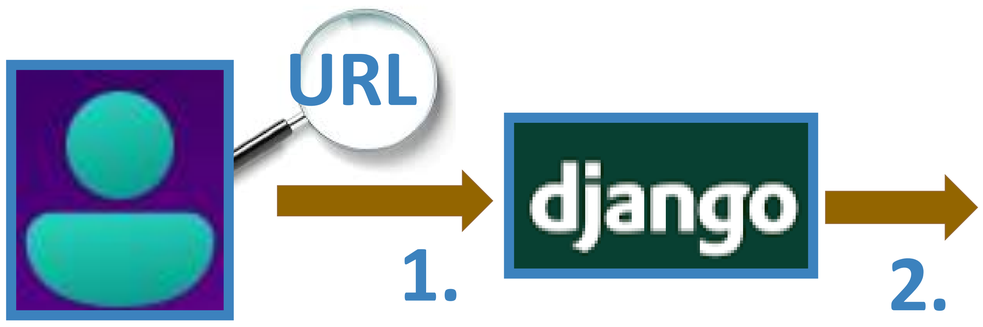
Interacción del navegante con su página web

- Los usuarios ingresan URLs en su navegador y envían solicitudes HTTP al servidor
- Django maneja estas solicitudes y las dirige a la vista correspondiente según las rutas URL definidas
- La vista procesa la solicitud, interactúa con el modelo y utiliza la plantilla para crear una respuesta HTML
- La respuesta HTML se envía al navegador del usuario, que la interpreta y muestra la página web

A continuación, vamos a ver el proceso por el cual el usuario es capaz de ver nuestra página web, una vez que ha sido subida a Internet

Proyecto en local

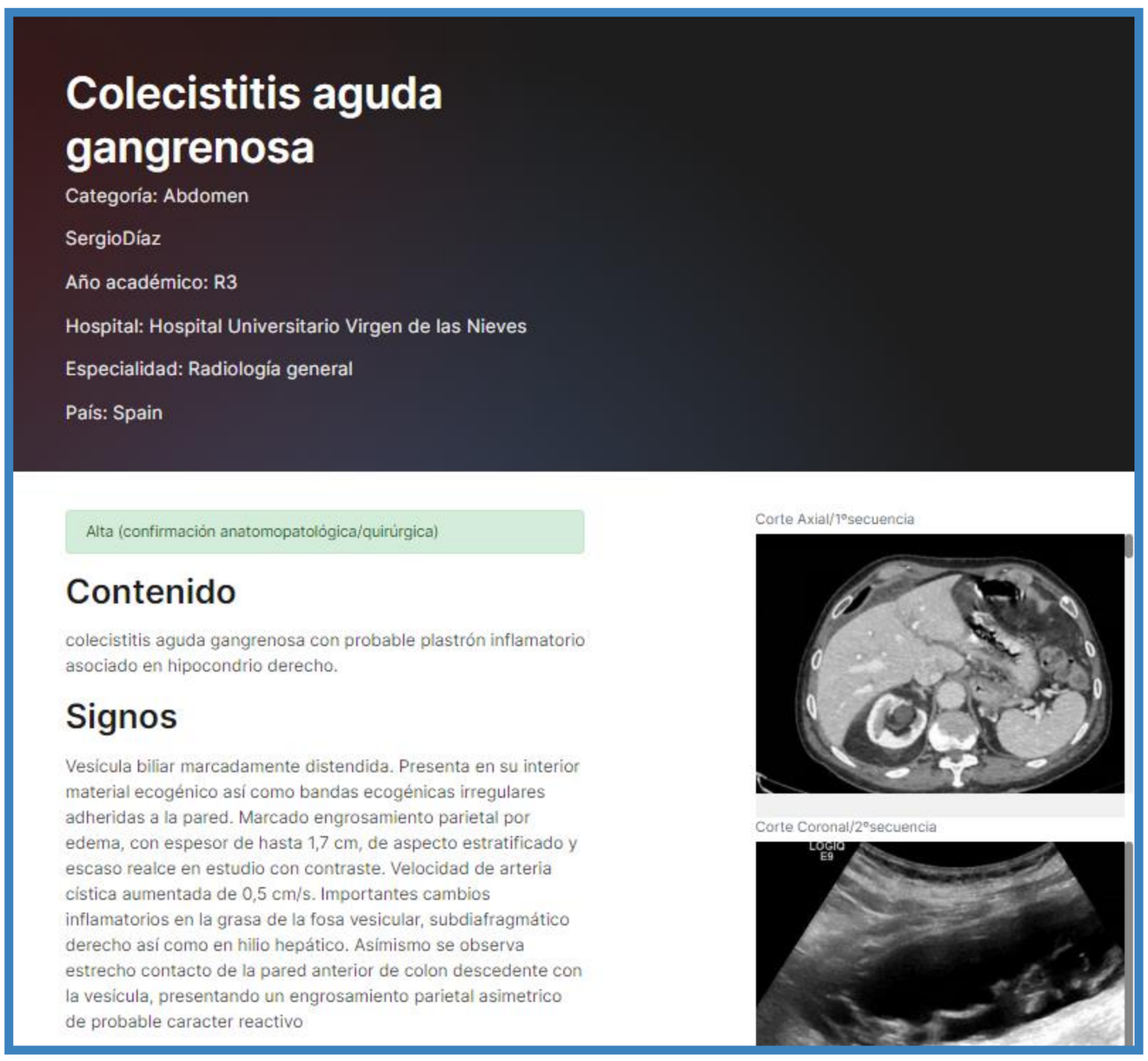
Diagrama resumen de los pasos anteriores




```
path('tc/patologia/abdomen/<int:post_id>/', views.blog_abdomentc,
```

```
def blog_abdomentc (request,post_id):
    posts= tcabdomen.objects.get(id=post_id)
    n= tcabdomen.objects.all()
    current_url= request.resolver_match.view_name

    return render (request, "entrada_tc.html", {'posts':posts
```

```
<main class="container">
  <div class="p-4 p-md-5 mb-4 text-white  bg-dark">
    <div class="col-md-6 px-0">
      <h1 class="display-4 fst-italic">{{posts.title}}</h1>
      <p class="lead my-3">{{posts.subtitulo}}</p>
      </div>
</div>
<div class="row mb-2">
  <div class="col-md-6">
    <div class="row g-0 border  overflow-hidden flex-md-row mb-4 sha
      <div class="col p-4 d-flex flex-column position-static">
        <h4 class="d-inline-block mb-2 text-primary">{{posts.autor}}
        <h4 class="mb-0" > {{posts.explicaautor}}</h4>
      </div>
      <div class="col-auto d-none d-lg-block">
        {% if posts.image %}
          <img src="{{posts.imagenautor.url}}" alt="Promoción"   wid
          {% endif %}
```



3.

1. *Un usuario busca «vesícula RadiologyModels»*
2. *Django procesa la solicitud de URL*
3. *La URL asociada es la vista blog_abdomentc y su plantilla HTML es entrada_tc.html*
4. *Se procesa la HTM para ser vista por el usuario*

Despliegue en Internet

¿Cómo mostramos nuestra página web al mundo?

Para que nuestra página web sea accesible a todo el público en general a través de Internet necesitamos utilizar un **PROVEEDOR DE HOSTING** (alojarla en servidor web) eligiendo un nombre de **DOMINIO** que no exista. Posteriormente configuraremos de forma sencilla el **DNS (traduce dominio a IP)** para que apunten a la dirección IP de nuestra página

Azure

aws

Analogía aclaratoria: *El servidor web sería como nuestra casa y el dominio como su dirección postal. El DNS sería el cartero que envía la solicitud para acceder a nuestra página. El servidor web sería la ciudad donde está la casa*

Despliegue en Internet

SUBIDA DEL PROYECTO A INTERNET; desarrollado previamente en entorno local. Realizamos algunas modificaciones básicas en seguridad, archivos estáticos y base de datos.

Pasos a seguir: El enrutamiento de URL conecta las solicitudes del usuario con las vistas adecuadas y con la base de datos recuperando los datos según sea necesario:

1. Configuración de la cuenta en proveedor de Hosting

2. Crear una base de datos SQL: Debemos de tomar nota de la conexión a la base de datos ya que lo necesitaremos más adelante

3. Nos aseguramos de que nuestro proyecto local funciona correctamente

4. Configuramos nuestro proyecto: Actualizamos el archivo *settings.py* con la base datos adquirida en el ***paso 2***

5. Fase de despliegue: Subimos el proyecto a un repositorio *Git* o similar. Mediante configuración pipeline de *CI/CD* enlazamos el código del repositorio al proyecto que hemos creado en nuestro proveedor de Hosting

6. Configuración de las variable de entorno: Configuramos las claves secretas de *Django* y la cadena de conexión a la base de datos *SQL* en las variables de entorno de tu aplicación

7. Despliegue de la aplicación: Ejecuta el pipeline de CI/CD para desplegar tu aplicación en *tu proveedor de hosting*

8. Actualiza las migraciones de la base de datos abriendo una terminal y ejecutando python *manage.py migrate* para aplicar las migraciones a la base de datos SQL en tu servidor

9. Verifica tu sitio web en línea: Abre tu sitio web en línea utilizando la URL proporcionada por *tu proveedor de hosting*

¿Qué es GitHub? GitHub

*Plataforma de desarrollo colaborativo y **de LIBRE ACCESO** de software que se utiliza para alojar y compartir proyectos con todo el mundo*

Explainable Intracranial Hemorrhage Detection for Radiologist Workflow Prioritization

Ejemplo maravilloso de un proyecto de detección automática de sangrado intracraneal (1) que podemos encontrar en Git

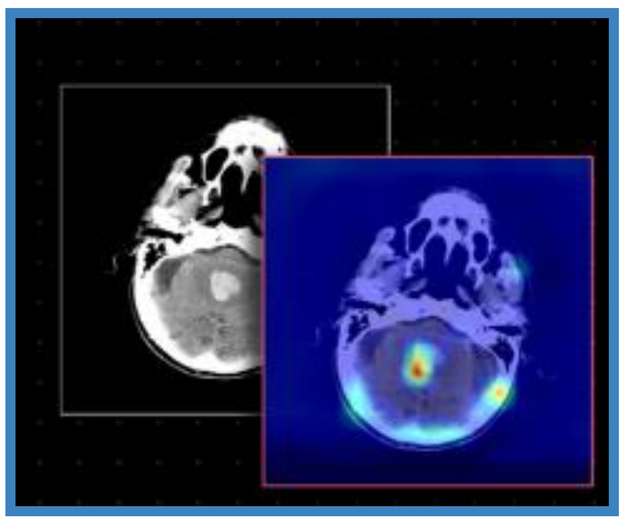


Imagen médica en recursos didácticos digitales

1. Imagen DICOM (Digital Imaging and Communications in Medicine)

El formato de imágenes DICOM es el sistema más utilizado para el almacenaje de imágenes médicas asociadas a metadatos.

VENTAJAS

- ***Metadatos asociados:*** Gran cantidad de metadatos asociados a la imagen facilitando seguimiento de la información del paciente
- **Estandarización**: Estándar en la industria médica permitiendo interoperabilidad de sistemas y dispositivos médicos
- **Flexibilidad**: DICOM admite una variedad de tipos de imágenes médicas
- ***Investigación: Permite el uso de*** herramientas para el procesamiento de imágenes médicas facilitando la aplicación de Inteligencia artificial
- **Cumplimiento Regulatorio**: Incluye medidas de seguridad intrínsecos que aumentan la protección de los datos

INCONVENIENTES

- ***NO SE PUEDEN MOSTRAR EN NAVEGADORES WEB → Necesitan previamente convertirse a formato JPG o similares (3)***
- ***Consumen gran cantidad de memoria***

Ninguna de las imágenes que vemos en Internet se encuentra en formato DICOM, primeramente han tenido que ser convertidas a otras imágenes

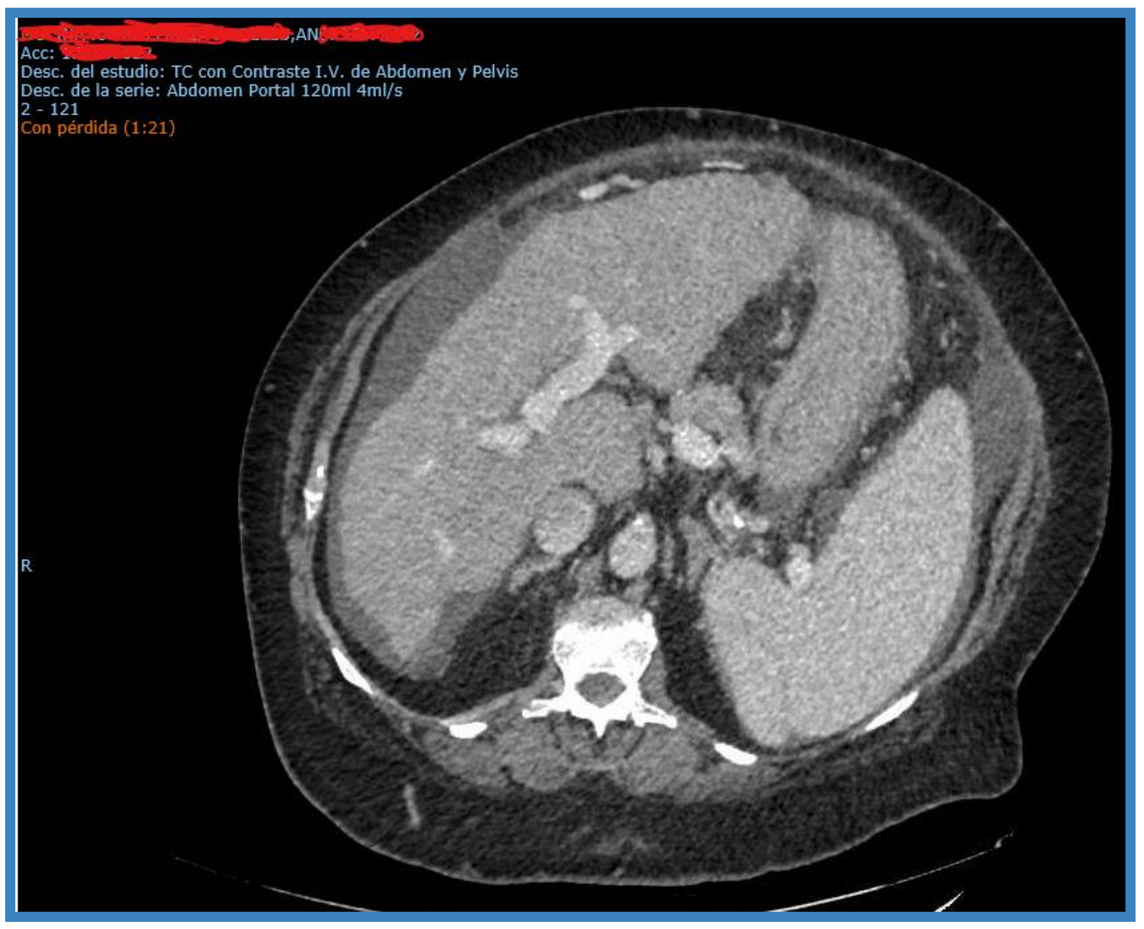
Imagen médica en recursos didácticos digitales

En el caso de que mostremos imágenes en formato DICOM es conveniente proceder a la eliminación del metadato asociado correspondiente a la información personal del paciente

Librerías en Python:

1. **Pydicom**: procesamiento de imágenes DICOM; incluyendo la posibilidad de su anonimización
2. **Numpy:** permite trabajar con los arrays de píxeles → aplicación en IA

```python
import pydicom
from pydicom.datadict import keyword_for_tag

def anonimizar_dicom(ruta_archivo_dicom, ruta_archivo_salida):
    # Cargar el archivo DICOM
    ds = pydicom.dcmread(ruta_archivo_dicom)

    # Lista de etiquetas a anonimizar (puedes agregar más según sea necesario)
    etiquetas_a_anonimizar = [
        'PatientName',
        'PatientID',
        'PatientBirthDate',
        'PatientSex',
        'PatientAddress',
        'OtherPatientIDs',
        'OtherPatientNames',
    ]

    # Anonimizar las etiquetas
    for etiqueta in etiquetas_a_anonimizar:
        if etiqueta in ds:
            ds.data_element(etiqueta).value = ''

    # Anonimizar cualquier otra información específica del paciente aquí
    # Por ejemplo, eliminar comentarios, descripciones, etc.

    # Guardar el archivo DICOM anonimizado
    ds.save_as(ruta_archivo_salida)

# Uso de la función
ruta_archivo_dicom = 'ruta/a/tu/archivo.dcm'
ruta_archivo_salida = 'ruta/a/tu/archivo_anonimizado.dcm'
anonimizar_dicom(ruta_archivo_dicom, ruta_archivo_salida)
```



Tras la implementación del código ya no aparecen datos identificativos

Ejemplo de código para eliminación de datos de paciente asociados a la imagen. Se sustituye la información por ‘ ’

Esto permite la anonimización de los datos como exige el Reglamento general de Protección de datos

Imagen médica en recursos didácticos digitales

2. Posibilidad de ver estudios **COMPLETOS** en páginas webs

Los recursos en Internet se han convertido en la fuente principal de información de los radiólogos para realizar consultas sobre los casos. Tenemos multitud de opciones…

seram Sociedad Española de Radiología Médica

RadioGraphics

…

No obstante, muchos veces echamos en falta… visualizar el estudio completo…. no podemos hacer *scroll* (desplazamiento vertical en las imágenes del estudio)…

¿Podemos añadir esta opción a nuestra página web?

Imagen médica en recursos didácticos digitales

2. Posibilidad de ver estudios **COMPLETOS** en páginas webs

A continuación vamos a ver como podríamos integrar en una página web cualquiera, un visor para visualización de estudios completos.

Una de las opciones más sencillas es utilizar un elemento ***scroll-container*** que consiste en un *box-element* que tiene en su interior "x" número de imágenes de tal forma que haciendo uso de ***JavaScript*** y ***CSS*** se pueda ajustar para apilar las imágenes una debajo de otra.

```
document.querySelectorAll('.scroll-container').forEach(function(scrollContainer) {
scrollContainer.addEventListener('scroll', function(e) {
  e.preventDefault();

  const scrollPosition = scrollContainer.scrollTop;
  const containerHeight = scrollContainer.clientHeight;
  const imageHeight = scrollContainer.querySelector('img').clientHeight;
  const diapositiva = (containerHeight / imageHeight);
  const imageCount = scrollContainer.querySelectorAll('img').length;
  const currentSlideIndex = Math.round(scrollPosition / imageHeight);

  let scrollTargetIndex = currentSlideIndex;

  // Ajustar la sensibilidad del desplazamiento en dispositivos móviles
  const sensitivityFactor = 2; // Aumentar la sensibilidad
```

Fragmento fundamental de código JavaScript que ejecuta este script cuando escucha la «llamada» de que el usuario quiere desplazarse por las imágenes

Barra lateral que permite el desplazamiento por todo el estudio

37 Congreso Nacional
CENTRO DE CONVENCIONES INTERNACIONALES
Barcelona 22/25 MAYO 2024
seram · FERM · RC RADIÓLEGS DE CATALUNYA

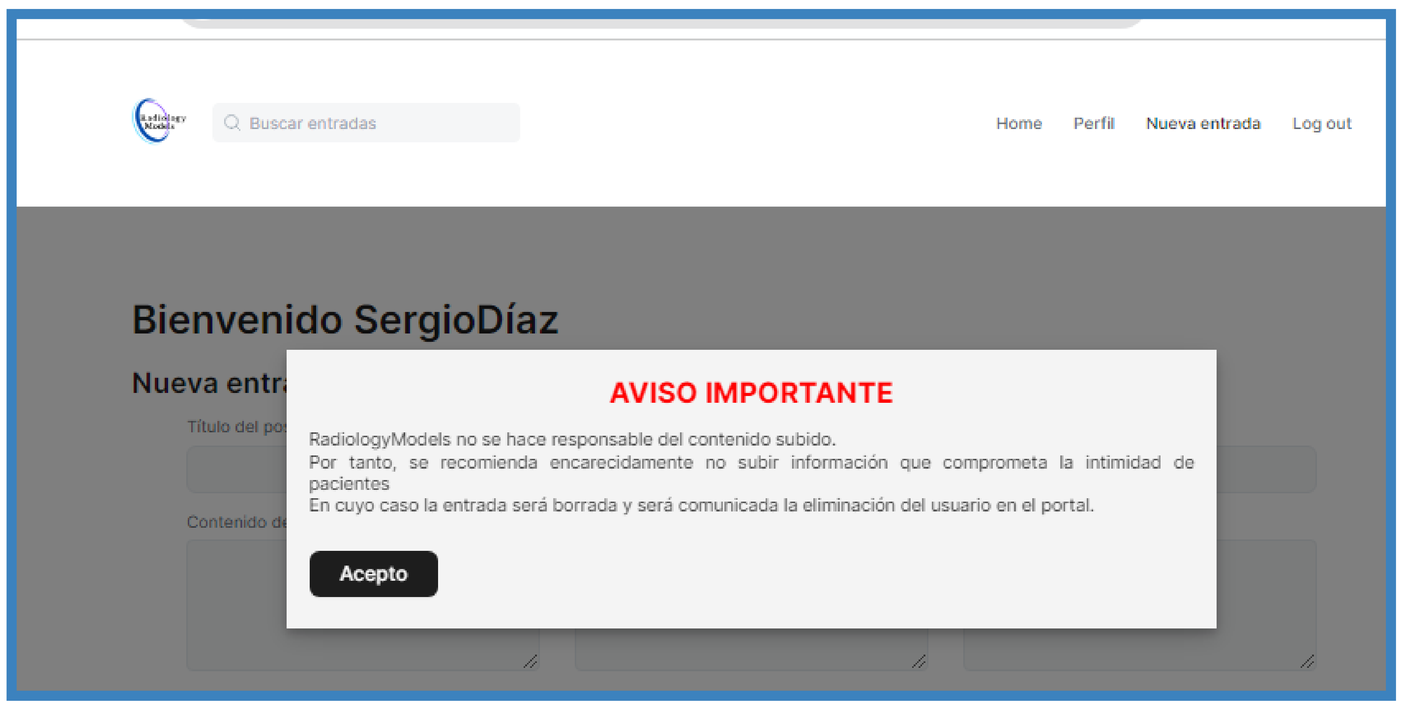
Imagen médica en recursos didácticos digitales

3. Prototipo de recogida de imágenes en web para su uso en IA

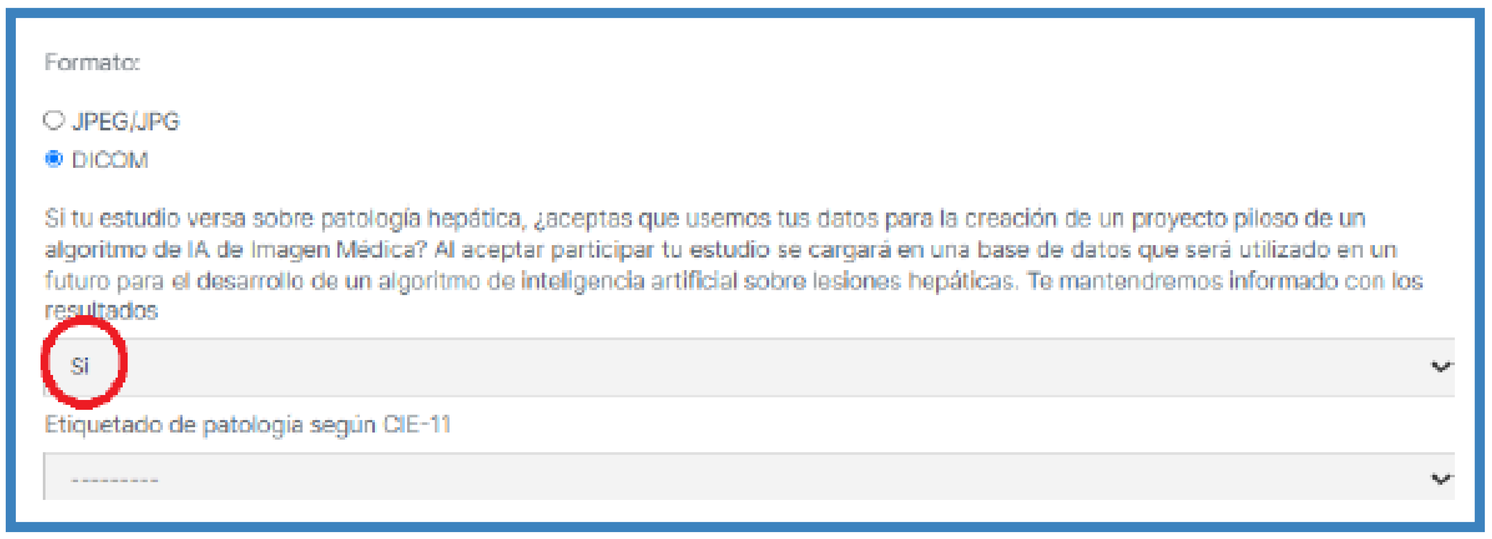
Ejemplo de pasos a seguir para generar una base de datos online que se utilice para generar un modelo de IA de patología hepática

Los principales obstáculos para creación de algoritmos de IA → recogida de datos (2)

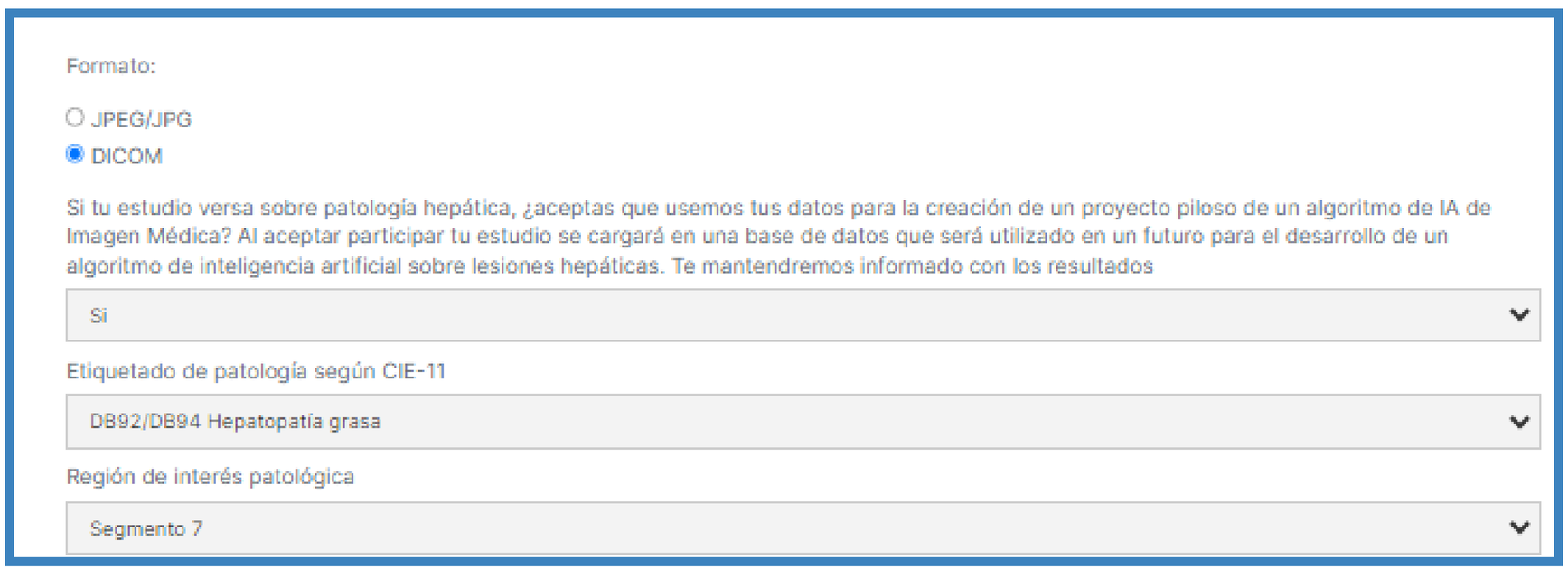
1. Tenemos en cuenta el **marco regulatorio** de protección de datos, avisando a los usuarios y **anonimizando estudios** como hemos visto anteriormente



2. Solicitamos **permiso al usuario** para introducirlo en la base de datos



3. Etiquetamos los datos; patología según **CIE-11 y localización** de la lesión



Resultados

Comprueba los resultados de este proyecto: Desarrollo web en Radiología, visor de imágenes y otras aplicaciones

¡¡Adentrémonos en el mundo de la programación informática!!

RadiologyModels

https://radiologymodels.com/

Página web de divulgación de contenido Radiológico HUVN, Granada

Portal

https://portal.radiologymodels.com/

Visor de estudios completos radiológicos

¿Tienes alguna idea de proyecto radiológico o quieres ver más ?

Visita → Radiology Models

Bibliografía

1. aws-samples. (2023). Radiology worklist ich detection [Código fuente]. GitHub. https://github.com/aws-samples/radiology-worklist-ich-detection

2. Willemink, M. J., Koszek, W., Hardell, C., Wu, J., Fleischmann, D., Harvey, H., Folio, L. R., Zhang, D., Rubin, D. L., & Lungren, M. P. (2020). Preparing medical imaging data for machine learning. Radiology, 295(1), 4-15. https://doi.org/10.1148/radiol.2020192224

3. Kahn, C. E., Langlotz, C. P., Channin, D. S., & Rubin, D. L. (2011). Informatics in Radiology: An information model of the DICOM Standard. RadioGraphics, 31(1), 295-304. https://doi.org/10.1148/rg.311105085.

4. RadiologyModels. (2022). Inicio. https://radiologymodels.com/