Evaluación N° 2 – Backend.

Fecha		**Sección / Curso**	
Nombre Estudiante		**RUT estudiante**	

Criterios de Evaluación – UNIDAD I

Instrucciones
La siguiente evaluación tiene como objetivo medir tu capacidad de desarrollar un proyecto con las herramientas vistas en clases referentes al uso de la herramienta Django y funcionalidades adicionales.

El sistema deberá implementar CRUDs para el modelo propuesto.

Criterio	Descripción	Puntaje Máx.
1. Entorno virtual "eva2"	Creación y uso correcto de entorno virtual.	2
2. Comentarios en bloque	Uso de comentarios explicativos en cada módulo o clase.	5
3. Estructura del proyecto	Organización de carpetas y archivos según buenas prácticas.	5
4. Modelo de datos	Creación de entidades y relaciones según requerimientos en una BD **PostgreSQL**.	10
5. CRUD completo por entidad	Implementación de Create, Read, Update, Delete para cada clase.	10
6. Carga BD con datos realistas	Existencia de datos de prueba coherentes.	5
7. Mejoras en el modelo	Inclusión de CHOICES y nuevas tablas adicionales.	6
8. Templates (vistas HTML)	Creación de páginas para CRUD fuera del admin.	10
9. Sistema de documentación	Implementación de documentación navegable de la API.	5
10. Filtros y búsquedas	Implementación de django-filters o búsqueda por campos clave.	5
11. Uso de PostgreSQL	Configuración y conexión correcta de la base de datos.	5
12. Rutas y endpoints de la API	Configuración de urls.py del proyecto y de la API.	5
13. Footer en templates	Inclusión de nombre, sección y año en el pie de página.	2
14. Nombrado del proyecto y la app	Nombres coherentes con la temática del sistema.	2

TOTAL 77 puntos

A continuación, se detalla el modelo base de la evaluación y sus instrucciones.

Escenario - Empresa de Salud Vital Ltda.

Proyecto de Backend - Contexto general

La empresa Salud Vital Ltda. es una clínica de tamaño mediano que busca digitalizar su sistema de administración de pacientes, médicos y atenciones médicas.

El nuevo sistema deberá registrar consultas, especialidades, tratamientos, recetas y medicamentos, permitiendo generar reportes filtrados por médico, especialidad o paciente.

El objetivo es desarrollar un **backend** completo usando Django REST Framework (DRF), con modelos interrelacionados, filtros y documentación.

Instrucciones de la EVALUACION.
La solución desarrollada por usted usando Django REST Framework (DRF) deberá tener las siguientes características:

Creación de un entorno virtual con el nombre de "eva2":
- Utilizar un entorno de desarrollo virtual con el nombre de "eva2".

Comentarios:
- El proyecto en su totalidad deberá contar con insertos de comentarios en el código en forma de bloques (no entre líneas).

Estructura:
- La estructura del proyecto deberá ser acorde a las vistas en clases pensando en mantener el buen orden de los elementos de un proyecto real.

Modelo de datos:
- A partir del modelo de datos propuesto usted deberá desarrollar y presentar una solución tal que cuente con las siguientes características:
    - Un CRUD para cada entidad del modelo.
    - Carga de datos en el modelo de forma que ya se encuentre con elementos de prueba (datos lo más cercano a la realidad).
- Existen varias opciones para poder mejorar el modelo ya sea con CHOICES o con la CREACIÓN DE NUEVAS TABLAS o entidades. Se deben crear 2 mejoras usando estas 2 opciones.

Vistas:
- El acceso a los datos deberá ser a través de una serie de TEMPLATES, para no tener que usar el "admin" del DRF.
- Crear las vistas para el CRUD (fuera del administrador) para cada una de las entidades (clases):
    - **Especialidad**: Crear, listar, actualizar, y eliminar.
    - **Paciente**: Crear, listar, actualizar, y eliminar.
    - **Medico**: Crear, listar, actualizar, y eliminar.
    - **consulta_medica**: Crear, listar, actualizar, y eliminar.
    - **tratamiento**: Crear, listar, actualizar, y eliminar.
    - **medicamento**: Crear, listar, actualizar, y eliminar.
    - **receta_medica**: Crear, listar, actualizar, y eliminar.

Documentación:
- Se requiere que se incluya un sistema de "Documentación" (ej: coreapi, openapi, swagger, etc.).

Filtros y búsqueda:
- Se requiere que se incluya un sistema de “Filtros” aplicados a campos clave del modelo (ej: médicos por especialidad, pacientes de cada médico, etc.).

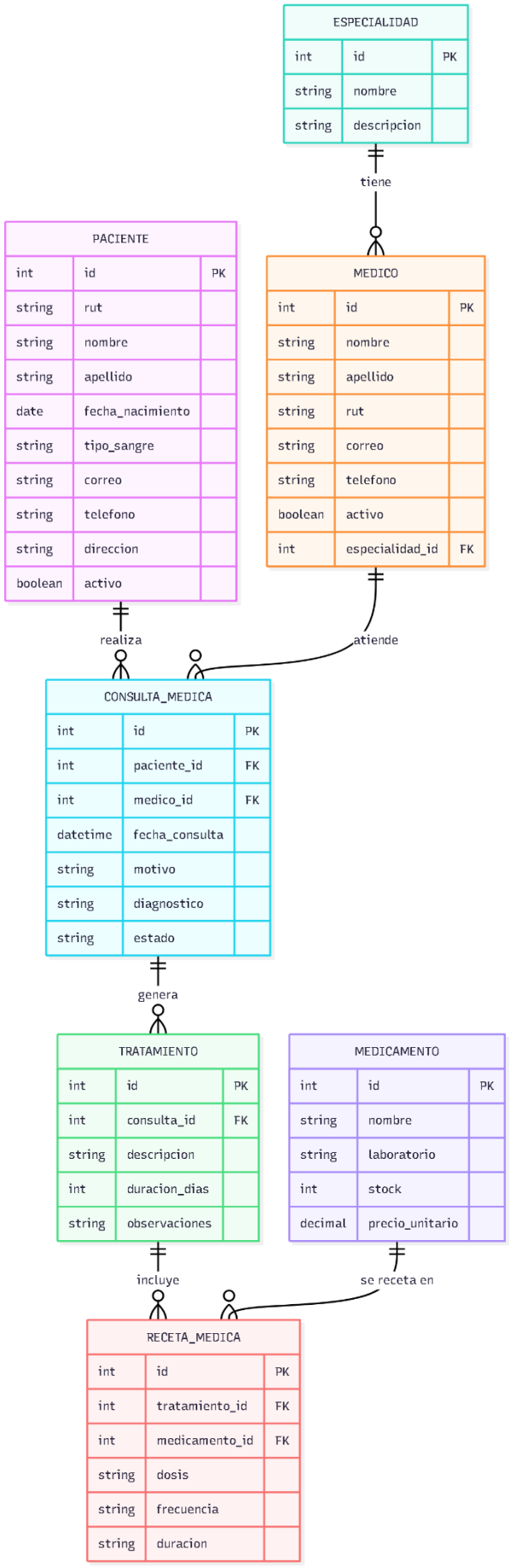
Choices:
- Se requiere que se incluya al menos una implementación de “CHOICES” en el modelo.

PostgreSQL:
- Se requiere que se use la base de datos PostgreSQL para este ejercicio.

Generales:
- Crear en el desarrollo un proyecto con un nombre adecuado a la temática del sistema.
- Crear en el desarrollo una API con un nombre adecuado a la temática del sistema.
- Deberá existir una ruta hacia el sistema “administrador” de DRF (no se usará).
- Deberá existir una ruta hacia los ENDPOINT de la API para cada clase.
- Las páginas (TEMPLATES) deberán tener su **nombre, sección y año** en el FOOTER (de todas las páginas).

Usted podrá ocupar como guía todos los documentos entregados en clases y todos los comodines disponibles.

Se apela a su criterio para realizar la prueba solos con la idea de medir su nivel de conocimiento en la asignatura.


ESPECIALIDAD
int id PK
string nombre
string descripcion
tiene
PACIENTE
int id PK
string rut
string nombre
string apellido
date fecha_nacimiento
string tipo_sangre
string correo
string telefono
string direccion
boolean activo
MEDICO
int id PK
string nombre
string apellido
string rut
string correo
string telefono
boolean activo
int especialidad_id FK
realiza
atiende
CONSULTA_MEDICA
int id PK
int paciente_id FK
int medico_id FK
datetime fecha_consulta
string motivo
string diagnostico
string estado
genera
TRATAMIENTO
int id PK
int consulta_id FK
string descripcion
int duracion_dias
string observaciones
MEDICAMENTO
int id PK
string nombre
string laboratorio
int stock
decimal precio_unitario
incluye
se receta en
RECETA_MEDICA
int id PK
int tratamiento_id FK
int medicamento_id FK
string dosis
string frecuencia
string duracion