ANEXO EXAMEN DE CERTIFICACIÓN

Plan de Estudio	Desarrollo Aplicaciones FullStack Python
	Caso Salas Operaciones

Caso “Salas Operaciones (Fullstack Python)”

La gerenta de una reputada clínica veterinaria, doña Ivonne Gonzale se encuentra con problemas para administrar las salas de operaciones del establecimiento que administra. Entre los problemas que encuentra es la descoordinación que tiene para asignar una sala de operaciones a los médicos veterinarios que trabajan con ella. Muchas confusiones en el horario de las salas han hecho concluir a Ivonne que es necesario un sistema que permita mantener una correcta organización y gestión de las salas en donde se realizan procedimientos quirúrgicos.

Luego de conocer los beneficios de desarrollar un sistema basado en el framework Django es que Ivonne le ha solicitado a Ud. que desarrolle un MVP (DEMO) de una aplicación. De esta manera, y en conjunto con el directorio de la clínica, podrán tomar la decisión de aprobar el desarrollo de un

sistema más complejo, que permita solucionar los problemas de gestión y administración de las salas de operaciones.

Es por esto que usted será responsable de elaborar una DEMO de la plataforma, de modo de convencer a Ivonne y al resto del equipo de contratar sus servicios como desarrollador(a) de software usando el stack Python/Django y el framework CSS Bootstrap. Desarrollando crear una aplicación Web escalable y robusta sin incurrir en excesivos tiempos de implementación.

Antecedentes

Luego de reunirse con Ivonne, actualmente se cuentan con los siguientes antecedentes:

- La clínica tiene actualmente 4 salas de operaciones. Pero puede ampliarse y sumar nuevas salas.
- Cada sala debe tener un código único, un tamaño (en metros cuadrados), y una breve descripción con el equipo que posee, y un color (código HEX)
- Los usuarios del sistema son los médicos veterinarios, deben ser capaces de hacer **Login** en el sistema, y **Registrarse**. Cada médico debe tener al menos un nombre, correo electrónico y una contraseña.
- Al momento de registrarse deben quedar como inactivos. Solo el **superuser** puede pasarlos al estado activo (considere el campo **is_active** del modelo User de Django-Auth).
- Para agendar un procedimiento se debe seleccionar una fecha, una hora de inicio (no se necesitan minutos), una hora de fin, una sala, una mascota y un doctor (quién agenda el procedimiento).
- Evidentemente un doctor no puede ocupar una sala que a esa hora está ocupada. Tampoco puede estar en dos procedimientos al mismo tiempo.
- La clínica tiene actualmente un sistema de fichas médicas de las mascotas. Para efectos de esta DEMO puede agregarlas usando Django-Admin.

Requerimientos Funcionales del Sistema

Después de varias reuniones entre Igor y nuestro equipo técnico, los requerimientos que pudimos bajar son los siguientes:

- Desarrollar una aplicación Web basada en el prototipo propuesto en las ilustraciones
- La aplicación debe poder navegarse desde la barra superior. En la misma debe saludarse al usuario por su nombre (si está logueado).
- Las funcionalidades descritas deben estar protegidas para que sean realizadas sólo por usuarios registrados y autentificados correctamente. En el caso de que un usuario no esté registrado, el sistema debe redirigirlo a la pantalla para que haga Login o se Registre.

- En la vista principal (ruta: **/**) se deben ver un calendario de la semana (los 7 días). En cada columna (día) se deben ver los procedimientos agendados, ordenados por hora. El color de fondo debe ser el de la sala agendada.
- En la vista antes mencionada debe poderse cancelar un procedimiento, pero sólo de lso procedimientos del que el médico logueado es el creador.
- Se debe tener un enlace al formulario para agendar (en ruta: **/procedimientos**).
- En caso de que el agendamiento de un procedimiento falle, se debe poder dar un feedback al médico.
- Los formularios pueden validarse en el FrontEnd o en el BackEnd. Esto queda a su discreción
- Se puede implementar la autenticación de usuarios usando Django-Auth o un sistema customizado. Pero si se debe procurar que las contraseñas estén encriptadas.
- Las rutas deben ser exactas a lo que muestran las maquetas. La excepción de esto son las rutas de autenticación. En el caso de que use otras rutas se debe especificar en el **README.txt**.

Requerimiento Final

Debe existir un botón que abra un modal (o página aparte) que muestre el bloque liberado de cualquier sala, más próximo, y un botón para agendarlo. Considere este el último requerimiento por desarrollar.

Maquetas (Mockups)

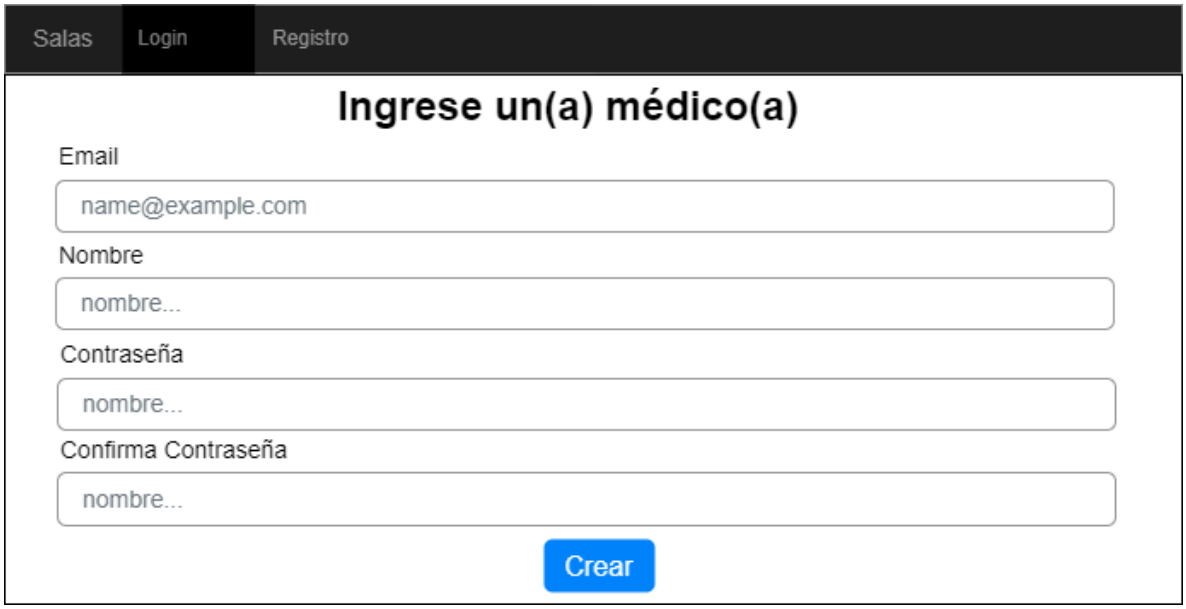


Ilustración 1: Pantalla de Registro

Ruta: /registro

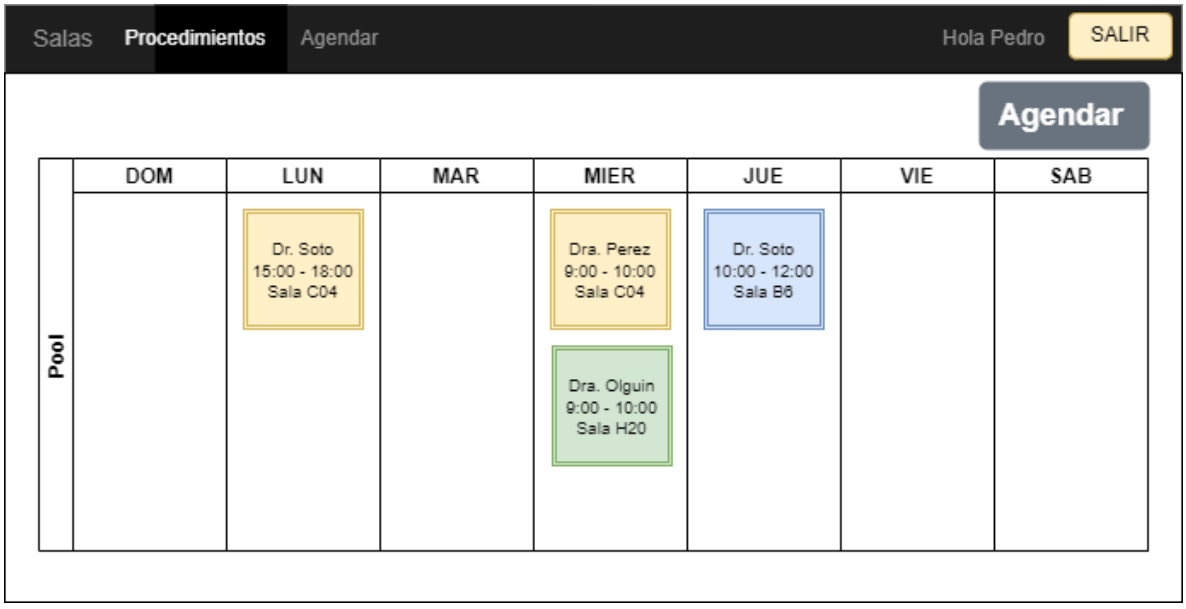


Ilustración 2: Vista Home

Ruta: /

Ilustración 3: Formulario de Agendamiento

Ruta: /procedimientos

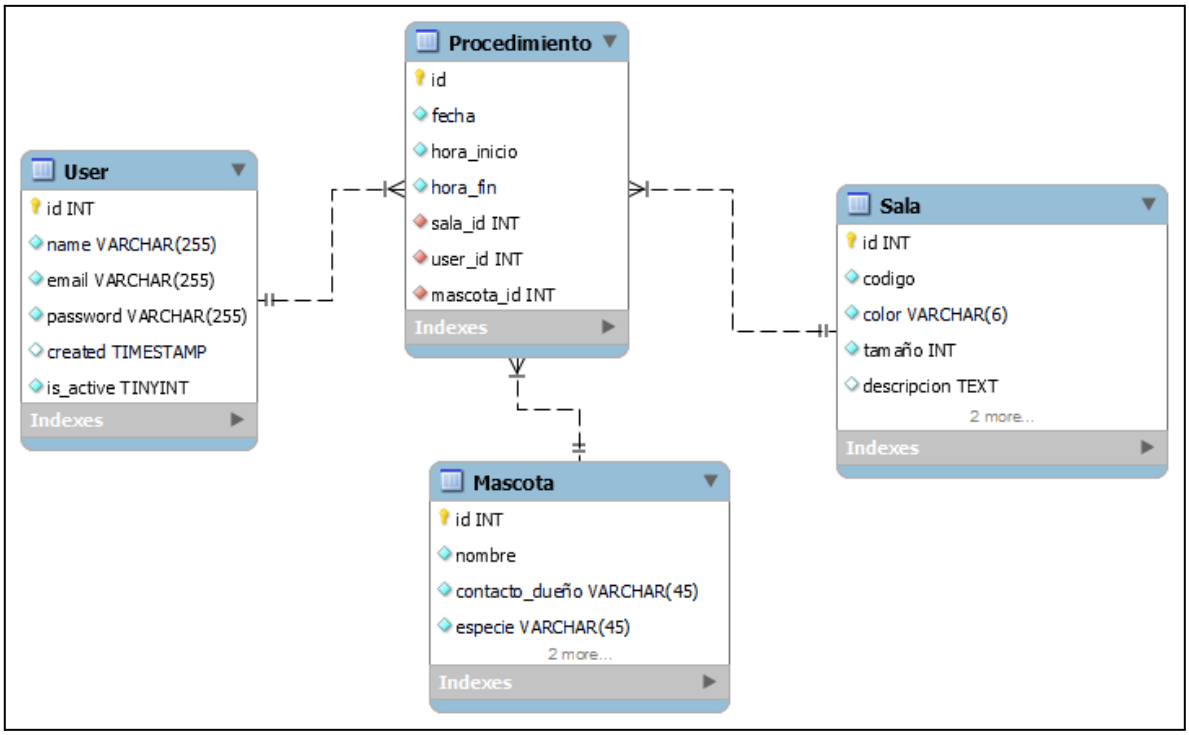
Requerimientos No Funcionales

El arquitecto del proyecto, le ha hecho las siguientes definiciones:

- El sistema debe utilizar una base de datos PostgreSql.
- El sistema debe construirse con Python/Django
- El sistema Web debe construirse bajo el patrón MVC.
- El acceso de usuarios se puede implementar con el modelo de Django Auth, o un modelo personalizado que usted prefiera.
- La capa de acceso a datos debe ser implementada con el ORM Django
- El archivo README.txt debe ir en la carpeta del proyecto.

Modelo de Datos

A continuación, se presenta el modelo de datos que ya ha sido diseñado por el equipo técnico del proyecto. Puede usarlo como guía para entender el problema, pero **no se preocupe** si los campos cambian de nombre al utilizar la Django ORM, o si desea quitar, agregar campos.



Ayuda

Puede serle útil la siguiente función:

```python
from datetime import timedelta, date

def get_weeks_days ():
  today =  date.today()
  days = []
  weekday = today.weekday()
  first_date = today - timedelta(days=weekday)
  for i in range(7):
    days.append(first_date + timedelta(days=i))

  return days


print(get_weeks_days())
```

Entregable

Haga todos los supuestos que estime conveniente para la resolución del caso y déjelos debidamente comentados en el código fuente en caso de requerirse. Recuerde, al final del proyecto entregar lo siguiente:

- Código fuente del proyecto
- El proyecto debe llamarse exactamente **salas_operaciones** (sin espacios ni mayúsculas, con guión bajo).
- Export de la base de datos con esquema y datos, (le puede consultar al instructor a cargo). Este modelo debe llamarse exactamente **salas_operaciones.sql** (sin espacios ni mayúsculas, con guión bajo), y debe ir dentro de la carpeta del proyecto. Este export debe contener al menos 5 equipos de al menos 2 categorías distintas, Un cliente, un operario, el superadmin.
- El username del superadmin de DjangoAdmin debe ser **admin**, y su contraseña debe ser **admin**. (en caso de no ser así, debe especificar el usuario y contraseña en un README.txt en la raíz del proyecto).
- Archivo **README.txt** con todas las indicaciones para ejecutar el proyecto. Debe contener las dependencias, y los datos de los usuarios.
- Queda estrictamente prohibido compartir el código en github u otra plataforma.
- No se puede utilizar ningún framework Frontend reactivo ni Docker