

47GIIN PROYECTO DE INGENIERÍA DEL SOFTWARE

Finalización de la iteración #01

Moisés Sevilla Corrales
PROF. Doctor Horacio Daniel Kuna





Índice

Finalización de la iteración #01	3	
Consigna:	3	
Enlaces de la actividad	<u>c</u>	



Finalización de la iteración #01

Consigna:

Documento con los detalles de avances logrados al final de la **iteración #01 del proyecto**.

Adicionalmente, se tendrá que incorporar qué ítems de trabajo se van a implementar en la siguiente iteración.

El formato va a ser adaptable según la metodología de gestión seleccionada.

En este caso se debe incluir el **video** de referencia de los avances según lo previsto por la asignatura (se recomienda que sea un enlace para visualizar el mismo que no requiere estar público).

En este documento se deberán incluir los enlaces a:

- Herramienta de gestión de incidencias.
- Herramienta de gestión de versionado de código.



- Creamos el Desarrollo inicial.
 - Vamos a usar la base de datos de PostgreSQL, con la interfaz de pgAdmin 4.
 - En el Backend usaremos Django REST framework para la comunicación con la base de datos y la creación de endpoints.
 - o En el Frontened usaremos Node.js con Vue.js.
 - o Para el desarrollo usaremos Windows 11 con Visual Studio Code.
- Empezamos creando la base de datos con un un script en pgAdmin.
 - Creamos la base de datos consultorio si no existe.
 - Definimos la tabla paciente.

```
-- Crear la base de datos para PostgreSQL
CREATE DATABASE IF NOT EXISTS consultorio;

-- Crear la tabla Paciente
CREATE TABLE Paciente (

id_paciente SERIAL PRIMARY KEY,
dni VARCHAR(9) UNIQUE, -- Nuevo campo DNI
nombre VARCHAR(100) NOT NULL,
apellido VARCHAR(100) NOT NULL,
email VARCHAR(100) UNIQUE NOT NULL,
telefono VARCHAR(15),
contrasena VARCHAR(100) NOT NULL

);
```

- Creamos un proyecto en Django, lo llamamos "consultorio".
 - Lo configuramos, creamos los endpoints e views.py para el paciente.

```
# CRUD Pacientes

class PacienteListCreateView(APIView):

GET: Lista todos los pacientes.

POST: Crea un nuevo paciente si no existe por id_paciente.

def get(self, request):

pacientes = Paciente.objects.all().order_by('id_paciente')

serializer = PacienteSerializer(pacientes, many=True)

return Response(serializer.data, status=status.HTTP_200_OK)
```



```
post(self, request):
        id_paciente = request.data.get('id_paciente')
       dni = request.data.get('dni')
       if id_paciente and Paciente.objects.filter(id_paciente=id_paciente).exists():
            return Response({"error": "El paciente ya existe."}, status=status.HTTP_400_BAD_REQUEST)
       if dni and Paciente.objects.filter(dni=dni).exists():
            return Response({"error": "El DNI ya está registrado."}, status=status.HTTP_400_BAD_REQUEST)
       serializer = PacienteSerializer(data=request.data)
       if serializer.is_valid():
           serializer.save()
           return Response(serializer.data, status=status.HTTP_201_CREATED)
       return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
class PacienteRetrieveUpdateDeleteView(APIView):
    GET: Recupera un paciente por id_paciente.
    PUT: Actualiza un paciente existente.
   DELETE: Elimina un paciente existente.
   def get(self, request, id_paciente):
           paciente = Paciente.objects.get(id_paciente=id_paciente)
       except Paciente.DoesNotExist:
           return Response({"error": "Paciente no encontrado."}, status=status.HTTP_404_NOT_FOUND)
       serializer = PacienteSerializer(paciente)
       return Response(serializer.data, status=status.HTTP_200_OK)
   def put(self, request, id_paciente):
           paciente = Paciente.objects.get(id_paciente=id_paciente)
       except Paciente.DoesNotExist:
           return Response({"error": "Paciente no encontrado."}, status=status.HTTP_404_NOT_FOUND)
       serializer = PacienteSerializer(paciente, data=request.data)
       if serializer.is valid():
           serializer.save()
           return Response(serializer.data, status=status.HTTP_200_OK)
       return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
   def delete(self, request, id_paciente):
            paciente = Paciente.objects.get(id_paciente=id_paciente)
       except Paciente.DoesNotExist:
           return Response({"error": "Paciente no encontrado."}, status=status.HTTP_404_NOT_FOUND)
       paciente.delete()
       return Response({"mensaje": "Paciente eliminado exitosamente."}, status=status.HTTP_200_OK)
```



Definimos el modelo de la base de datos en el backend en models.py

```
# Modelo para los pacientes

class Paciente(models.Model):

id_paciente = models.AutoField(primary_key=True) # Identificador único del paciente

dni = models.CharField(unique=True, max_length=9) # DNI del paciente

nombre = models.CharField(max_length=100) # Nombre del paciente

apellido = models.CharField(max_length=100) # Apellido del paciente

email = models.CharField(unique=True, max_length=100) # Correo único del paciente

telefono = models.CharField(max_length=15, blank=True, null=True) # Teléfono del paciente (opcional)

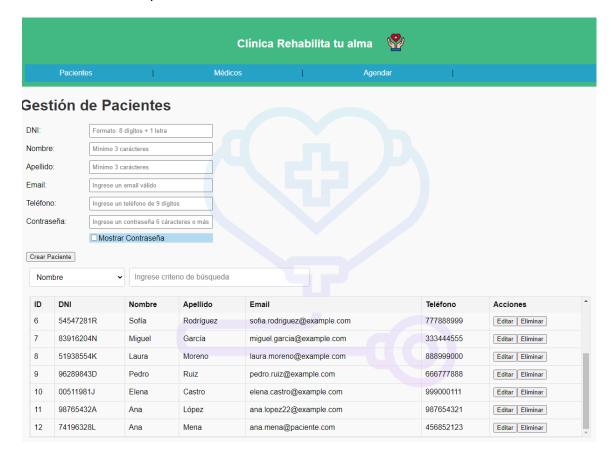
contrasena = models.CharField(max_length=100) # Contraseña del paciente

class Meta:

managed = False

db_table = 'paciente'
```

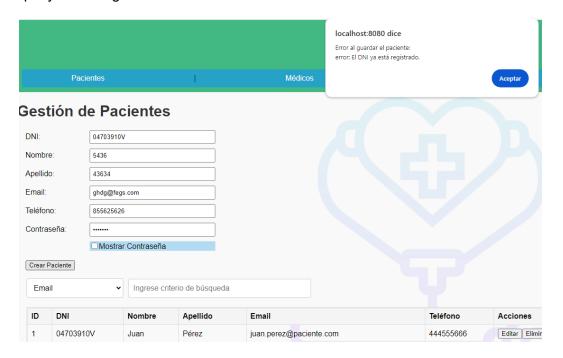
 Luego en la parte del frontend en VUE.js trans configurarlo y cumplir todos los puntos de la iteración nos queda esta interfaz.



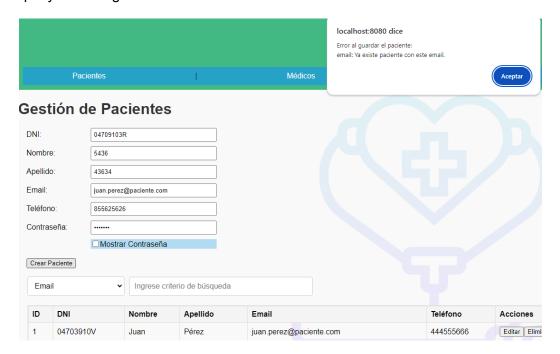
 Controlamos todos los campos del formulario para validarlos correctamente antes de crear el paciente, mostramos un CRUD completo del paciente en vista del modo de desarrollador.



 El DNI es único para cada paciente, no se puede repetir, nos mostrara una alerta de que ya está registrado.

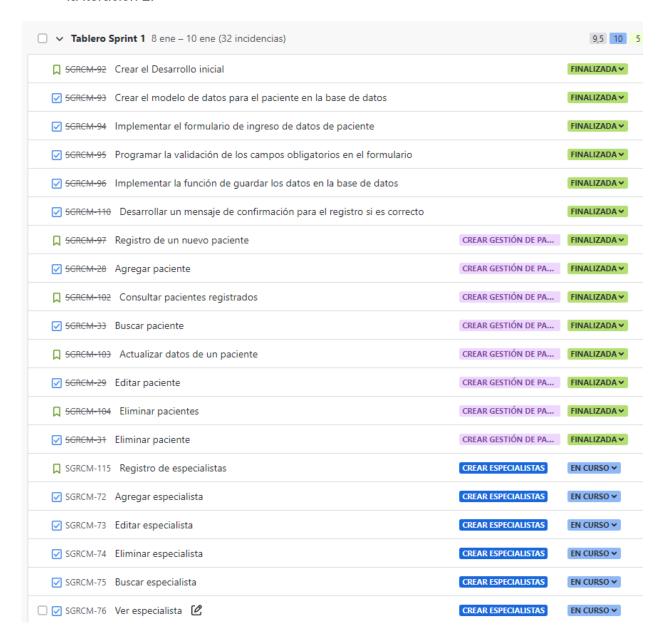


 El email es único para cada paciente, no se puede repetir, nos mostrara una alerta de que ya está registrado.





 Vemos el Sprint 1 que teníamos definido y finalizamos la iteración 1, para continuar con la iteración 2.





Enlaces de la actividad

Video de presentación de este entregable (YouTube):

URL: https://youtu.be/t45tf4bSV-4

Herramienta de gestión de incidencias (Jira):

URL: https://moisessevilla.atlassian.net/jira/software/projects/SGRCM/boards/1/backlog?selecte
dlssue=SGRCM-92

Herramienta de gestión de versionado de código (GitHub):

URL:https://github.com/moisessevilla/47GIIN Proyecto Consultorio