

[VIEW.PY](#)

A continuación se detallan los bloques lógicos más relevantes de los controladores secundarios.

BLOQUE REGISTRO:

Ubicacion: Funcion kairos_register.

Este bloque contiene el proceso necesario para que el usuario pueda crear su cuenta, esto por medio de un formulario en el cual el usuario envía sus datos para posteriormente ser almacenados en la base de datos.

#Recibimos los datos por medio del formulario, estos datos seran los correspondientes a las metricas del usuario.

```
username = request.POST['user']

email = request.POST['email']

password = request.POST['password']

peso = float(request.POST.get('peso', 0) or 0)

altura = float(request.POST.get('altura', 0) or 0)

edad = int(request.POST.get('edad', 0) or 0)

nivel_texto = request.POST.get('nivel')

objetivo_texto = request.POST.get('objetivo')

deporte_texto = request.POST.get('deporte')
```

#Verificamos que el correo no este en uso

```
if User.objects.filter(email=email).exists():

    messages.info(request, 'El correo ya está en uso')

    return redirect('register')

else:

    try:

        with transaction.atomic():
```

#Creamos el usuario y almacenamos sus datos

```
        user = User.objects.create_user(username=username, email=email,
password=password)
```

```
        user.save()
```

#Creamos un espacio en la base de datos para almacenar las metricas fisicas y la configuracion del usuario

```
        MetricasFisicas.objects.create(
```

```
            usuario=user,
```

```
            peso=peso,
```

```
            altura=altura,
```

```
            edad=edad,
```

```
            nivel=nivel_texto,
```

```
            objetivo=objetivo_texto,
```

```
            deporte=deporte_texto
```

```
        )
```

```
        Configuracion.objects.create(
```

```
            usuario=user,
```

```
            nivel_psico=4,
```

```
            modo_psicologico=1,
```

```
            modo_competencia=False
```

```
        )
```

#Enviar al usuario a la intrefaz principal, de otra forma mostrar mensaje de error

```
        messages.success(request, 'Registro exitoso')
```

```
        return redirect('login')
```

```
except Exception as e:
```

```
    print(f"Error en registro: {e}")
```

```
    messages.error(request, 'Error al crear el perfil.')
```

```
    return redirect('register')
```

else:

#Enviar los datos predefinidos a la interfaz de registro.

niveles = MetricasFisicas.NIVELES

objetivos = MetricasFisicas.OBJETIVOS

deportes = MetricasFisicas.DEPORTES

return render(request, 'kairos_register.html', {

 'niveles': niveles,

 'objetivos': objetivos,

 'deportes': deportes

})

BLOQUE LOGIN:

Ubicacion: Funcion kairos_login.

Este bloque contiene el proceso necesario para que el usuario pueda acceder a su cuenta (creada previamente en la interfaz de registro), accediendo a la base datos, y verificando los datos por medio de un formulario.

#Recibimos los datos por medio del formulario

```
user_input = request.POST['user']

password = request.POST['password']

username_login = user_input
```

#Dependiendo de si `user_input` tiene `@`, podremos verificar si es un correo o un nombre de usuario.

```
if '@' in user_input:

    try:

        user_obj = User.objects.get(email=user_input)

        username_login = user_obj.username

    except User.DoesNotExist:

        username_login = None
```

#Realizamos la autenticación, si el usuario existe enviamos al usuario a la interfaz principal con su cuenta activa, de otra forma enviaremos un mensaje de alerta.

```
user = None

if username_login:

    user = auth.authenticate(username=username_login, password=password)

    if user is not None:

        auth.login(request, user)

        return redirect('main')

    else:

        messages.info(request, 'Usuario, Correo o Contraseña inválida')
```

```
return redirect('login')
```

[MODELS.PY](#)

A continuación se detallan los bloques lógicos más relevantes del archivo [models.py](#), este archivo contiene el código encargado de definir la estructura de los datos contenidos en nuestro proyecto, tomando la información y generando una base de datos con estos mismos.

Básicamente nos permite gestionar las características de la base de datos, esto utilizando clases y objetos.

BLOQUE 1: CONSTRUCCIÓN DE LA TABLA “MetricasFisicas”

Ubicación: Clase MetricasFisicas.

En este bloque creamos la Tabla dedicada a almacenar los datos de usuario, más específicamente sus métricas físicas. El API debe ser capaz de recibir estos datos, provocando que la IA posea un comportamiento dinámico.

CÓDIGO COMENTADO:

#Creamos una serie de conjuntos, mismos que poseen elementos que servirán para crear valores predefinidos para ciertas entradas del usuario.

```
NIVELES = [
```

```
    ('Bajo', 'Bajo'),
```

```
    ('Medio', 'Medio'),
```

```
    ('Alto', 'Alto'),
```

```
]
```

```
OBJETIVOS = [
```

```
    ('Aumentar Músculo', 'Aumentar Músculo'),
```

```
    ('Definición', 'Definición'),
```

```
    ('Mejorar Resistencia', 'Mejorar Resistencia'),
```

```
    ('Aumentar Fuerza', 'Aumentar Fuerza'),
```

```
    ('Bienestar', 'Bienestar'),
```

```
]
```

```
DEPORTES = [  
    ('Calistenia', 'Calistenia'),  
    ('Fuerza/Gimnasio', 'Fuerza/Gimnasio'),  
    ('Resistencia/Cardio', 'Resistencia/Cardio'),  
    ('Fitness funcional', 'Fitness funcional'),  
    ('Combate/Artes Marciales', 'Combate/Artes Marciales'),  
    ('Movilidad/Mente', 'Movilidad/Mente'),  
]
```

#Definimos los tipos de datos que tendrá la base de datos.

#Cada uno de los campos posee sus propias características y limitaciones, de lo más destacable sería el usuario, mismo que toma como base a la sección `User`, misma que viene predefinida en Django.

#El atributo `CASCADE` se utiliza para que al momento de eliminar un objeto, todos los objetos que poseen el mismo valor sean eliminados respectivamente

```
usuario = models.OneToOneField(User, on_delete=models.CASCADE)  
  
peso = models.DecimalField(max_digits=5, decimal_places=2, help_text="Peso en Kg")  
  
altura = models.DecimalField(max_digits=5, decimal_places=2, help_text="Altura en Metros")  
  
edad = models.IntegerField()  
  
nivel = models.CharField(max_length=30, choices=NIVELES, default='Bajo',  
verbose_name="Nivel de Experiencia")  
  
objetivo = models.CharField(max_length=30, choices=OBJETIVOS, default='Bienestar',  
verbose_name="Objetivo Principal")  
  
deporte = models.CharField(max_length=30, choices=DEPORTES, default='Fitness funcional',  
verbose_name="Deporte Principal")
```

#Define un mensaje para que se muestre en la interfaz que muestra Django para visualizar la base de datos.

```
def __str__(self):  
    return f"Métricas de {self.usuario.username}"
```

BLOQUE 2: CONSTRUCCIÓN DE LA TABLA "Configuracion"

Ubicación: Clase Configuracion.

En este bloque creamos la Tabla dedicada a almacenar los datos de la configuración del usuario, estos datos funcionarán para que la API cambie su comportamiento dependiendo de los datos.

CÓDIGO COMENTADO:

#Al igual que en la clase anterior debemos crear un conjunto, este conjunto almacena datos predefinidos para determinar el `modo_psicologico` del usuario.

```
ARQUETIPOS = [  
    (1, 'Empático'),  
    (2, 'Técnico'),  
    (3, 'Élite'),  
]
```

#Definimos los tipos de datos que tendrá la base de datos.

#Lo más destacable sería la segunda aparición del atributo `CASCADE`, teniendo el mismo propósito que en la clase anterior.

```
usuario = models.OneToOneField(User, on_delete=models.CASCADE)  
  
nivel_psico = models.IntegerField(default=5, validators=[MinValueValidator(1),  
MaxValueValidator(9)], verbose_name="Nivel Psico")  
  
modo_psicologico = models.IntegerField(choices=ARQUETIPOS, default=1 ,  
verbose_name="Modo Psicológico")  
  
modo_competencia = models.BooleanField(default=False, verbose_name="Modo  
Competencia")
```

```
def __str__(self):  
    return f"Configuración de {self.usuario.username}"
```

BLOQUE 3: CONSTRUCCIÓN DE LA TABLA "Historial"

Ubicación: Clase Historial.

En este bloque creamos la Tabla dedicada a almacenar los mensajes escritos por el usuario, de igual manera almacena la respuesta de la IA y la fecha en la cual se envió el mensaje.

En cuanto a su estructura, es bastante simple y no posee nada especial al compararla con las otras tablas.

CÓDIGO COMENTADO:

#Esta clase a diferencia de las anteriores no posee conjuntos con valores predefinidos.

```
usuario = models.ForeignKey(User, on_delete=models.CASCADE)

fecha = models.DateTimeField(auto_now_add=True)

mensaje_usuario = models.TextField(verbose_name="Pregunta del Usuario")

respuesta_ia = models.TextField(verbose_name="Respuesta de KAIROS")


def __str__(self):

    return f"Chat de {self.usuario.username} - {self.fecha.strftime('%d/%m/%Y')}}"
```

¿Qué es la Inteligencia Artificial?

¿Para qué se ocupa la IA hoy en día?

Menciona los principales temas de estudio de la Inteligencia Artificial.

¿Todas las IA son comerciales o cualquier persona puede desarrollar una?

¿Te interesa algún tema en particular relacionado a la IA? ¿Cuál?

Menciona los lenguajes de programación más asociados con la IA.