PRACTICA #6

Andrés Gallegos | 30.127.148

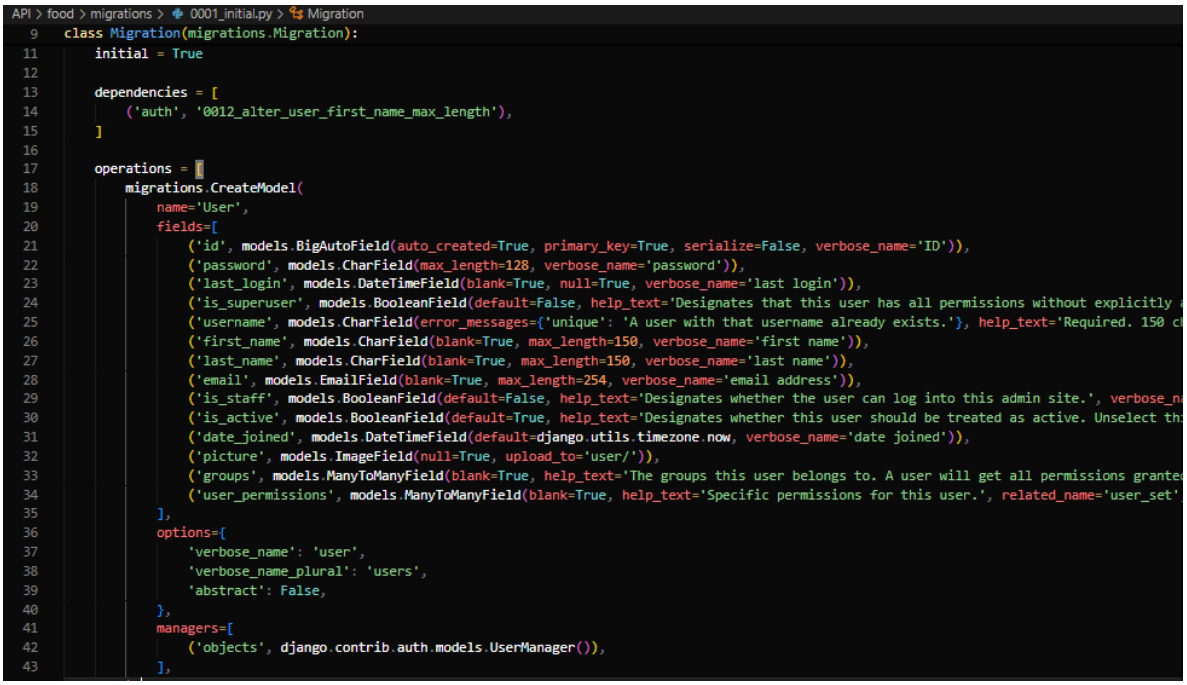
María Chang | 28.083.608

Moisés Londoño | 29.750.712

Luís Amias | 30.292.216

Prof. Venecia Miranda

CUMPLIMIENTO DE PRINCIPIOS SOLID:



*El modelo **User** tiene una única responsabilidad la cual es representar a un usuario en la base de datos cumpliendo así el **SPR**, el modelo está diseñado para ser extensible sin modificar la estructura existente, es decir, se cumple con el **OCP** y se puede sustituir sin problema cumpliendo así con el **LSP**.*

```python
class Migration(migrations.Migration):

    dependencies = [
        ('food', '0001_initial'),
    ]

    operations = [
        migrations.CreateModel(
            name='Recipe',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('name', models.CharField(max_length=255, unique=True)),
                ('instruction', models.TextField()),
                ('description', models.TextField()),
                ('type', models.CharField(max_length=255)),
            ],
        ),
        migrations.CreateModel(
            name='Food',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('name', models.CharField(max_length=50)),
                ('type', models.CharField(max_length=50)),
                ('quantity', models.PositiveIntegerField()),
                ('entry_date', models.DateField()),
                ('consume_date', models.DateField(blank=True, null=True)),
                ('expiration_date', models.DateField(blank=True, null=True)),
                ('its_bad', models.BooleanField(default=False)),
                ('user', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to=settings.AUTH_USER_MODEL)),
            ],
        ),
        migrations.CreateModel(
            name='RecipeIngredients',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('quantity', models.PositiveIntegerField()),
                ('food', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='food.food')),
                ('recipe', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='food.recipe')),
            ],
```

*Acá hemos definido tres modelos **Recipe** (representa una receta), **Food** (representa un alimento) y **RecipeIngredients** (Gestiona la relación receta/ingrediente), como podemos observar cada modelo tiene una única responsabilidad claramente definida cumpliendo asi con el **SRP**, los modelos son extensibles sin necesidad de modificar la estructura existente cumplen con el **OCP** y las dependencias son comunes y no problemáticas cumpliendo el **DIP**.*

```python
class Migration(migrations.Migration):

    operations = [
        migrations.CreateModel(
            name='Category',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('name', models.CharField(max_length=50)),
            ],
        ),
        migrations.RemoveField(
            model_name='recipeingredients',
            name='recipe',
        ),
        migrations.RemoveField(
            model_name='recipeingredients',
            name='food',
        ),
        migrations.RemoveField(
            model_name='food',
            name='consume_date',
        ),
        migrations.RemoveField(
            model_name='food',
            name='entry_date',
        ),
        migrations.RemoveField(
            model_name='food',
            name='its_bad',
        ),
        migrations.RemoveField(
            model_name='food',
            name='type',
        ),
        migrations.AddField(
            model_name='food',
            name='category',
            field=models.ForeignKey(default=1, on_delete=django.db.models.deletion.CASCADE, to='food.category'),
            preserve_default=False,
        ),
```

```
        migrations.CreateModel(
            name='Entry_History',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('entry_date', models.DateField(auto_now_add=True)),
                ('entry_quantity', models.PositiveIntegerField()),
                ('food', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='food.food')),
            ],
        ),
        migrations.CreateModel(
            name='Output_History',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('output_date', models.DateField(auto_now_add=True)),
                ('output_quantity', models.PositiveIntegerField()),
                ('motive', models.CharField(max_length=12)),
                ('food', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='food.food')),
            ],
        ),
        migrations.DeleteModel(
            name='Recipe',
        ),
        migrations.DeleteModel(
            name='RecipeIngredients',
        ),
    ]

```

*Acá podemos ver que los modelos **Category** (representa una categoría de alimentos), **Entry_History** y **Output_History** (registran las entradas y salidas de alimentos, cada uno se ocupa de un aspecto específico del historial de alimentos) es decir, tienen responsabilidades claras y especificas cumpliendo así con el **SRP**, tampoco tenemos métodos innecesarios así que se cumple **ISP** y las dependencias son comunes y no problemáticas cumpliendo con el **DIP**.*

```
class User(AbstractUser):
    picture = models.ImageField(upload_to='user/', null=True)

class Category(models.Model):
    name = models.CharField(max_length=50)

class Food(models.Model):
    name = models.CharField(max_length=50)
    quantity = models.PositiveIntegerField()
    expiration_date = models.DateField(null=True, blank=True)
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    category = models.ForeignKey(Category, on_delete=models.CASCADE)

class Entry_History(models.Model):
    food = models.ForeignKey(Food, on_delete=models.CASCADE)
    entry_date = models.DateField(auto_now_add=True)
    entry_quantity = models.PositiveIntegerField()

class Output_History(models.Model):
    food = models.ForeignKey(Food, on_delete=models.CASCADE)
    output_date = models.DateField(auto_now_add=True)
    output_quantity = models.PositiveIntegerField()
    motive = models.CharField(max_length=12)
```

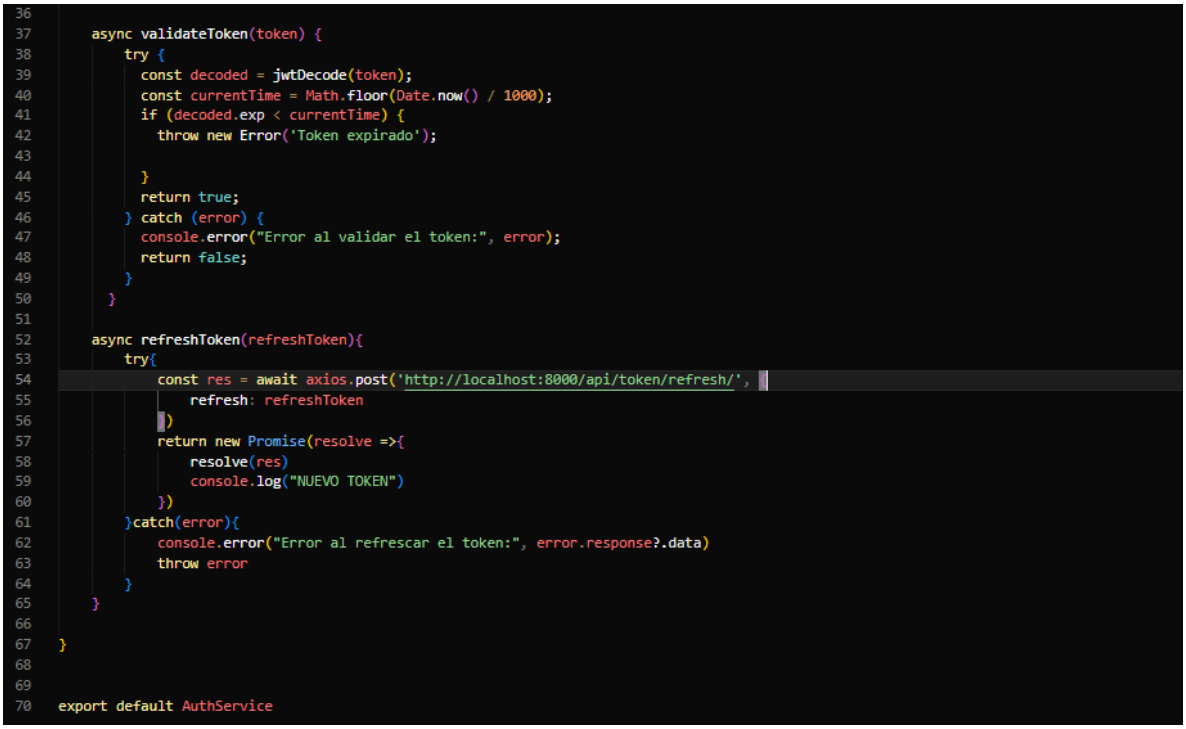
*Acá también vemos que cada modelo tiene una responsabilidad clara, **User** (Representa a un usuario), **Category** (Representa una categoría de alimentos), **Food** (representa un*

alimento y su relación con un usuario y una categoría), ***Entry_History*** *(Registra la entrada de alimentos) y* ***Output_History*** *(Registra la salida de alimentos) es decir se cumple* ***SPR****, también estos modelos pueden extenderse sin necesidad de modificar las clases existentes cumpliendo con el* ***OCP*** *y son simples y específicos por ello también se cumple* ***ISP****.*

```python
class UserSerializer(serializers.ModelSerializer):
    picture = serializers.ImageField(required=False)
    class Meta:
        model = User
        fields = ['username', 'email', 'first_name', 'picture', 'password']


class FoodSerializer(serializers.ModelSerializer):
    class Meta:
        model = Food
        fields = '__all__'

class Entry_HistorySerializer(serializers.ModelSerializer):
    class Meta:
        model = Entry_History
        fields = '__all__'

class Output_HistorySerializer(serializers.ModelSerializer):
    class Meta:
        model = Output_History
        fields = '__all__'

class CategorySerializer(serializers.ModelSerializer):
    class Meta:
        model = Category
        fields = '__all__'
```

Vemos que cada uno de los ***serializer*** *(UserSerializer, FoodSerializer, Entry_HistorySerializer, Output_HistorySerializer, CategorySerializer) tienen una única responsabilidad es decir cumplen con el* ***SRP****, los mismos están diseñados para ser fácilmente extensibles así que se cumple con el* ***OCP*** *y cada uno está específicamente diseñado para su modelo correspondiente sin tener métodos adicionales o campos que no estén relacionados con el modelo, si cumple* ***ISP****.*

```
class AuthService{
    constructor(){}
    async register(data){
        try{

            const res = await axios.post('http://localhost:8000/api/register/',data, {
                headers:{
                    'Content-Type': 'multipart/form-data'
                }
            })
            return new Promise(resolve =>{
                resolve(res)
            })
        }catch(error){
            console.error("Error al registrar:", error.response?.data)
            throw error
        }
    }

    async login(username, password){
        try{
            const res = await axios.post('http://localhost:8000/api/token/', {
                username: username.value,
                password: password.value
            })
            return new Promise(resolve =>{
                resolve(res)
            })
        }catch(error){
           console.log(error.response)
        }
    }
```



```

    async validateToken(token) {
        try {
          const decoded = jwtDecode(token);
          const currentTime = Math.floor(Date.now() / 1000);
          if (decoded.exp < currentTime) {
            throw new Error('Token expirado');

          }
          return true;
        } catch (error) {
          console.error("Error al validar el token:", error);
          return false;
        }
      }

    async refreshToken(refreshToken){
        try{
            const res = await axios.post('http://localhost:8000/api/token/refresh/', {
                refresh: refreshToken
            })
            return new Promise(resolve =>{
                resolve(res)
                console.log("NUEVO TOKEN")
            })
        }catch(error){
            console.error("Error al refrescar el token:", error.response?.data)
            throw error
        }
    }

}


export default AuthService
```

*Nos encontramos ahora en el **front**, analizando el **AuthService** que es un servicio que hace las peticiones, (todas las peticiones de la app se encuentran aquí) y se llaman desde cualquier parte, se está consciente de que en un futuro se pudiera realizar una separación más estricta de la clase, sin embargo el hecho de tener un servicio centralizado que maneje todas las peticiones relacionadas con la autenticación es una ventaja ya que facilita la*

gestión y el mantenimiento del código, simplificando así la forma en la que otros componentes interactúan con la lógica de autenticación haciendo que el código sea más sencillo de entender, reduciendo la posibilidad de errores e inconsistencias al realizar configuraciones en múltiples archivos o clases.