Matricula Estudiante (1) - [enrollStudent]

? Descripción

*Este servicio **matricula automáticamente a un estudiante** en todos los programas de capacitación correspondientes a su **puesto actual (designation)**.*

El flujo toma como origen:

- **Student** (por DNI)
- **Program**
- **Course**
- **Program Enrollment**

*El servicio determina **qué cursos pertenecen al puesto del estudiante**, identifica **qué programas contienen esos cursos** y finalmente crea **matrículas individuales** para cada programa en el ERP.*

? Endpoint

POST `/enroll-student`

? Parámetros (Request Body)

```
{
  "dni": "string"
}
```

Campo	Tipo	Obligatorio	Descripción
dni	string	Sí	DNI del estudiante para buscarlo en el ERP

? Seguridad

Requiere autenticación interna mediante:

- `dbErp()` para consultas SQL
- `ServiceErp()` para consumo REST de recursos del ERP

? Flujo del Servicio (resumen)

1. Validar parámetro `dni`

Si no se envía DNI:
→ retorna error inmediato.

---

2. Buscar al estudiante en `tabStudent`

```
SELECT name, puesto, joining_date
FROM tabStudent
WHERE dni = <dni>
ORDER BY joining_date DESC
LIMIT 1
```

Si no existe:
→ retorna "*no se encontró al estudiante*".

---

3. Obtener todos los Programas del ERP

GET `/resource/Program?fields=["name"]&limit=None`

Por cada programa:

- Se consulta `/resource/Program/<program_name>`
- Se leen los cursos incluidos en ese programa
- Se construye el mapa:

`curso → [programas donde aparece]`

4. Obtener todos los Cursos del ERP

GET `/resource/Course?fields=["name"]`

Por cada curso:

- Se consulta `/resource/Course/<course_name>`
- Se revisan los `designation` asociados al curso
- Se construye el mapa:

`puesto → [cursos habilitados para ese puesto]`

---

5. Validar si el puesto del estudiante tiene cursos asignados

Si no tiene cursos asociados:
→ retorna "*No se encontró programas para este puesto*".

---

6. Determinar los programas finales

Para cada curso del puesto, buscar en qué programas aparece y construir la lista final:

`programas_finales = unique(programas_permitidos)`

Si la lista queda vacía:
→ no existen programas asignados a este puesto.

---

7. Registrar las matrículas (Program Enrollment)

Para cada programa identificado:

1. Validar si ya existe matrícula:

```
GET Program Enrollment?filters=[
    ["student","=",<id_student>],
    ["program","=",<program>]
]
```

2. Si **existe**, se almacena como encontrado.

3. Si **no existe**, se crea:

```
POST Program Enrollment
{
  "student": "<student_id>",
  "program": "<program>",
  "enrollment_date": "<fecha_hoy>",
  "academic_year": "<año>",
  "docstatus": 1
}
```

El servicio separa:

- Matrículas creadas
- Matrículas ya existentes
- Errores durante el registro

? Respuesta 200 – Ejemplo

```
{
  "valor": true,
  "msn": "Lista de programas",
  "data": [
    "PROG-001",
    "PROG-005"
  ],
  "enrollment": [
    { "name": "ENR-0001" },
    { "name": "ENR-0002" }
  ],
  "errors": []
}
```

? Posibles Errores

1. Falta DNI

```
{
  "valor": false,
  "msn": "Falta enviar \"student\""
}
```

2. Estudiante no encontrado

```
{
  "valor": false,
  "msn": "no se encontró al estudiante"
}
```

3. Puesto sin cursos asociados

```
{
  "valor": false,
  "msn": "No se Encontró programas para este puesto"
}
```

4. Error al registrar un enrollment

Se devuelve en `errors`:

```
{
  "errors": [
    "mensaje del error interno"
  ]
}
```

? Esquemas usados en el ERP

Student

```
{
  "name": "string",
  "dni": "string",
  "puesto": "string"
}
```

Program

```
{
  "name": "string",
  "courses": [
    { "course": "string" }
  ]
}
```

Course

```
{
  "name": "string",
  "designation": [
    { "designation": "string" }
  ]
}
```

Program Enrollment

```
{
  "student": "string",
  "program": "string",
  "enrollment_date": "date",
  "academic_year": "string",
  "docstatus": 1
}
```

? Lógica en Pseudocódigo

```
dni = request.dni

if empty(dni):
    return error

student = query Student where dni=dni order by joining_date desc limit 1
if not found:
    return error

designation = student.puesto

programas = GET all programs
mapProgramas = generar mapa curso → programas

cursos = GET all courses
mapCursos = generar mapa puesto → cursos

if designation not in mapCursos:
    return error

programas_final = programas donde cursos del puesto están incluidos

foreach programa in programas_final:
    if enrollment existe:
        agregar a enrollments encontrados
    else:
        crear enrollment
        agregar respuesta a enrollments
```

Revisión #1
Creado 2025-11-25 16:48:30 -05 por Elian

Actualizado 2025-11-25 17:02:45 -05 por Elian