# BENEMERITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

## Facultad de Ciencias de la Computación

## Modelos de desarrollo web

Actividad 2

Alumna: Liseth Danae Mecatl Toxcoyoa

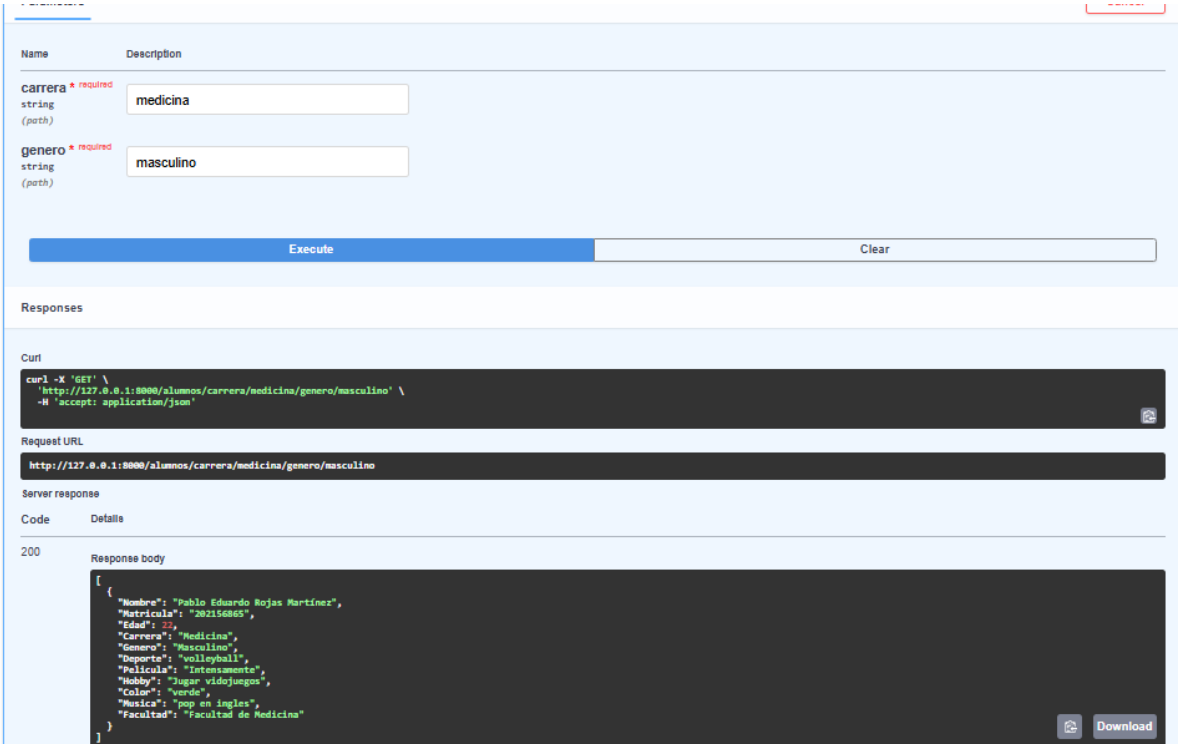
Matricula: 202233510

Fecha:19-jun-2025

# Endpoints PATH

## GET por Carrera y Género

```
@app.get("/alumnos/carrera/{carrera}/genero/{genero}")
async def get_alumnos_por_carrera_y_genero(carrera: str, genero: str):
    alumnos_filtrados = filter(
        lambda alumno: (carrera.lower() in alumno.Carrera.lower()) and
        (alumno.Genero.lower() == genero.lower()),
        alumnos
    )
    return list(alumnos_filtrados)
```




## GET por Facultad y Deporte

```
@app.get("/alumnos/facultad/{facultad}/deporte/{deporte}")
async def get_alumnos_por_facultad_y_deporte(facultad: str, deporte: str):
```

```
  alumnos_filtrados = filter(
    lambda alumno: (facultad.lower() in alumno.Facultad.lower()) and
    (deporte.lower() in alumno.Deporte.lower()),
    alumnos
  )
  return list(alumnos_filtrados)
```

### GET por Matrícula

```
@app.get("/alumnos/matricula/{matricula}")
async def get_alumno_por_matricula(matricula: str):
  alumnos_filtrados = filter(
    lambda alumno: alumno.Matricula == matricula, alumnos)
  try:
    return list(alumnos_filtrados)[0]
  except:
    return {"error": "No se ha encontrado el alumno con esa matrícula"}
```

### GET por Nombre (búsqueda exacta)

```
@app.get("/alumnos/nombre/{nombre}")
async def get_alumno_por_nombre(nombre: str):
  alumnos_filtrados = filter(
    lambda alumno: alumno.Nombre.lower() == nombre.lower(), alumnos)
  try:
    return list(alumnos_filtrados)[0]
  except:
    return {"error": "No se ha encontrado el alumno con ese nombre exacto"}
```

### GET por Carrera

```
@app.get("/alumnos/carrera/{carrera}")
async def get_alumnos_por_carrera(carrera: str):
    alumnos_filtrados = filter(
        lambda alumno: carrera.lower() in alumno.Carrera.lower(), alumnos)
    return list(alumnos_filtrados)
```

### GET por Edad

```
@app.get("/alumnos/edad/{edad}")
async def get_alumnos_por_edad(edad: int):
    alumnos_filtrados = filter(lambda alumno: alumno.Edad == edad, alumnos)
    return list(alumnos_filtrados)
```

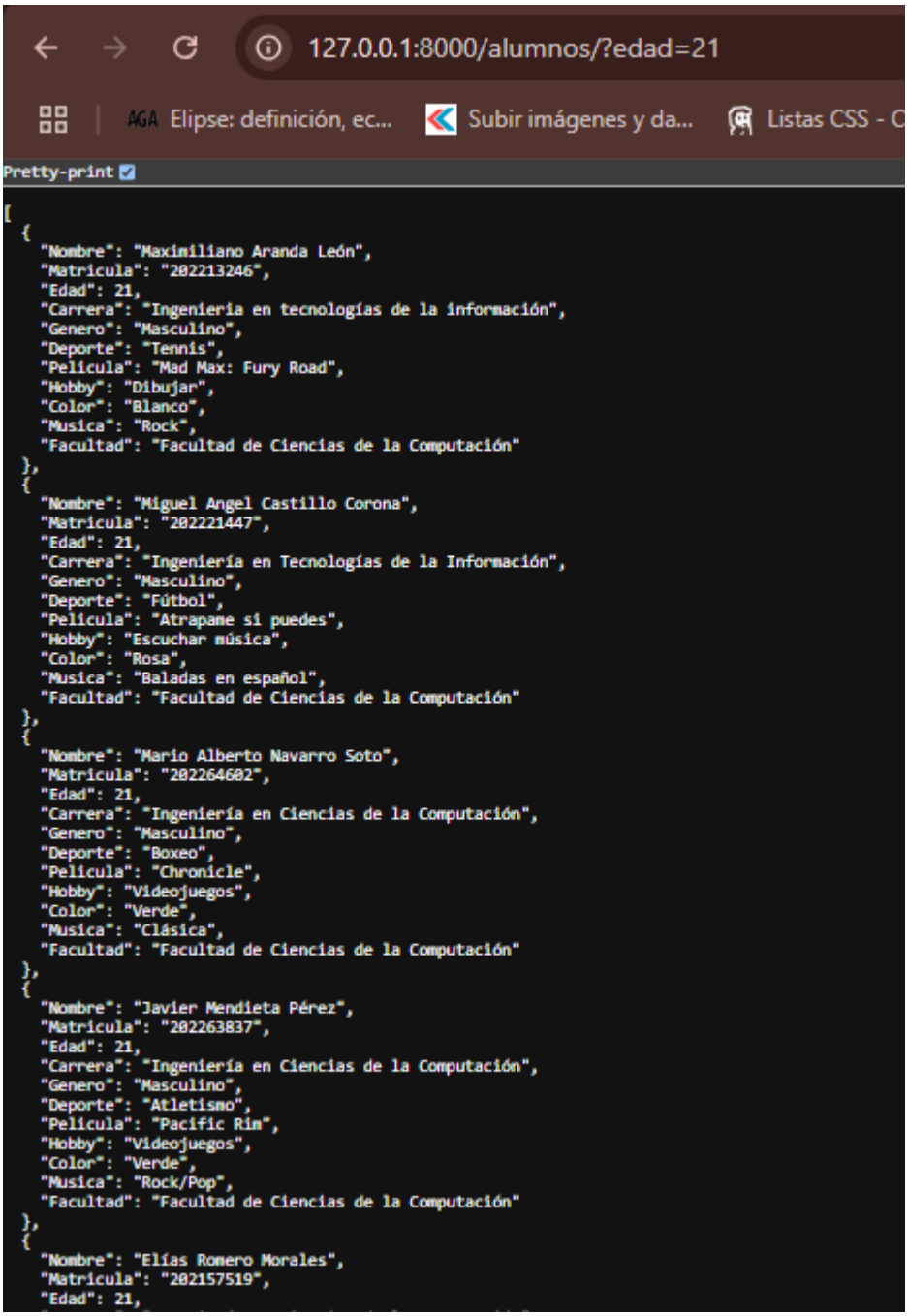
# Endpoints query

### GET con filtros combinados por query parameters

```
@app.get("/alumnos/")
async def filtrar_alumnos(
    edad: Optional[int] = None,
    carrera: Optional[str] = None,
    genero: Optional[str] = None,
    facultad: Optional[str] = None
):
    alumnos_filtrados = alumnos
```

```
if edad:
    alumnos_filtrados = filter(
        lambda alumno: alumno.Edad == edad, alumnos_filtrados)
if carrera:
    alumnos_filtrados = filter(lambda alumno: carrera.lower(
    ) in alumno.Carrera.lower(), alumnos_filtrados)
if genero:
    alumnos_filtrados = filter(
        lambda alumno: alumno.Genero.lower() == genero.lower(), alumnos_filtrados)
if facultad:
    alumnos_filtrados = filter(lambda alumno: facultad.lower(
    ) in alumno.Facultad.lower(), alumnos_filtrados)

return list(alumnos_filtrados)
```




## GET por deporte y música

```
@app.get("/alumnos/intereses/")
async def alumnos_por_intereses(
    deporte: Optional[str] = None,
    musica: Optional[str] = None
```

```
):
    alumnos_filtrados = alumnos

    if deporte:
        alumnos_filtrados = filter(lambda alumno: deporte.lower(
        ) in alumno.Deporte.lower(), alumnos_filtrados)
    if musica:
        alumnos_filtrados = filter(lambda alumno: musica.lower(
        ) in alumno.Musica.lower(), alumnos_filtrados)

    result = list(alumnos_filtrados)
    return result if result else {"message": "No se encontraron alumnos con esos criterios"}
```

## GET por rango de edades

```
@app.get("/alumnos/rango-edad/")
async def alumnos_por_rango_edad(
    edad_min: int,
    edad_max: int
):
    alumnos_filtrados = filter(
        lambda alumno: edad_min <= alumno.Edad <= edad_max,
        alumnos
    )
    return list(alumnos_filtrados)

# GET por hobby y color favorito

@app.get("/alumnos/hobbies/")
async def alumnos_por_hobby_color(
```

```
    hobby: Optional[str] = None,
    color: Optional[str] = None
):
    alumnos_filtrados = alumnos

    if hobby:
        alumnos_filtrados = filter(
            lambda alumno: hobby.lower() in alumno.Hobby.lower(), alumnos_filtrados)
    if color:
        alumnos_filtrados = filter(
            lambda alumno: color.lower() in alumno.Color.lower(), alumnos_filtrados)

    return list(alumnos_filtrados)
```

## GET por película favorita y facultad

```
@app.get("/alumnos/cine/")
async def alumnos_por_pelicula_facultad(
    pelicula: str,
    facultad: Optional[str] = None
):
    alumnos_filtrados = filter(
        lambda alumno: pelicula.lower() in alumno.Pelicula.lower(),
        alumnos
    )
    if facultad:
        alumnos_filtrados = filter(
            lambda alumno: facultad.lower() in alumno.Facultad.lower(),
            alumnos_filtrados
```

```
    )
result = list(alumnos_filtrados)
return result if result else {"message": "No se encontraron coincidencias"}
```