Instrucciones para ejecutar los tests del proyecto

El proyecto incluye dos conjuntos de pruebas:

- **Unit Tests** → tests/ProviderOptimizer.UnitTests
- **Integration Tests** → tests/ProviderOptimizer.IntegrationTests

Los tests están desarrollados en **xUnit**, utilizando el patrón AAA, mocks cuando es necesario y pruebas de integración contra PostgreSQL (vía Docker).

1. Prerrequisitos

Antes de ejecutar las pruebas, asegúrate de tener instalado:

- ✓ **.NET SDK 8.0**
- ✓ **Docker Desktop** (solo requerido para tests de integración)
- ✓ **PostgreSQL** (si no deseas usar Docker)

Puedes verificar con:

dotnet --version

docker --version

2. Ejecutar *TODOS* los tests

Desde la raíz del repositorio:

dotnet test

Esto compila las soluciones, restaura paquetes y corre todas las pruebas de manera automática.

3. Ejecutar solo los Unit Tests

dotnet test tests/ProviderOptimizer.UnitTests/ProviderOptimizer.UnitTests.csproj

Los tests unitarios validan:

- Lógica del servicio ProviderOptimizerService
- Selección del proveedor óptimo
- Filtrado por rating
- Manejo de escenarios límite (sin proveedores, valores inválidos, etc.)

No requieren base de datos ni Docker.

4. Ejecutar solo los Integration Tests

dotnet test tests/ProviderOptimizer.IntegrationTests/ProviderOptimizer.IntegrationTests.csproj

Los tests de integración validan:

- Acceso real a PostgreSQL vía conexión Npgsql
- Queries del ProviderRepository
- Integración con el OptimizerDbContext
- Datos iniciales cargados en la BD usando el script migrations/init.sql

Importante para las pruebas de integración:

Debes tener PostgreSQL levantado con Docker:

docker-compose up -d db

Esto inicia la BD, crea la base provider_optimizer y carga los datos iniciales.

5. Ejecutar tests en modo Verbose (con detalles)

dotnet test -v detailed

Útil para debugging cuando un test falla.

6. Ejecutar tests con cobertura (coverage)

Si deseas obtener métricas de cobertura:

dotnet test --collect:"XPlat Code Coverage"

El reporte se genera en:

TestResults/<GUID>/coverage.cobertura.xml

Puede visualizarse usando herramientas como:

- Coverlet
- ReportGenerator
- SonarQube

7. Limpiar y recompilar antes de ejecutar

En caso de errores de cache:

dotnet clean

dotnet build

dotnet test

8. Dentro de Docker (opcional)

Si deseas ejecutar los tests en contenedores:

docker-compose run provideroptimizer dotnet test

Aunque normalmente las pruebas se ejecutan fuera de la imagen para más flexibilidad.

9. Ubicación de los archivos de prueba

tests/

├── ProviderOptimizer.UnitTests/

│ ├── ProviderOptimizerTests.cs

│ └── ProviderOptimizer.UnitTests.csproj

│

└── ProviderOptimizer.IntegrationTests/

├── OptimizerIntegrationTests.cs

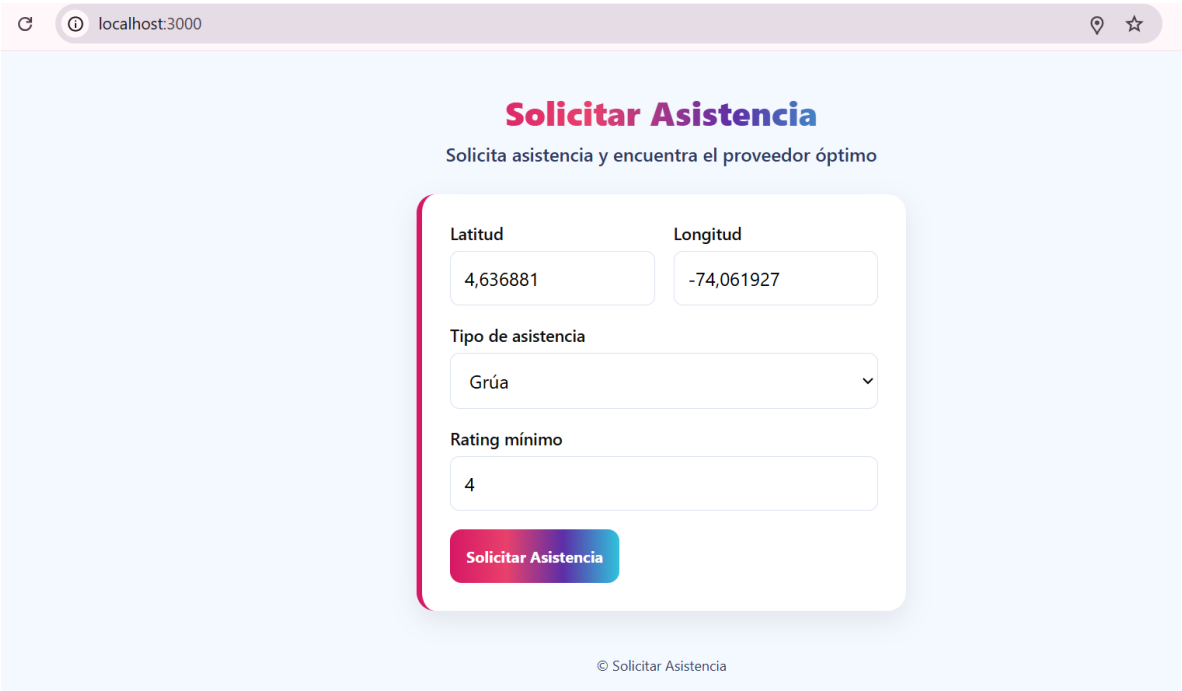
└── ProviderOptimizer.IntegrationTests.csproj

Resumen para entregar

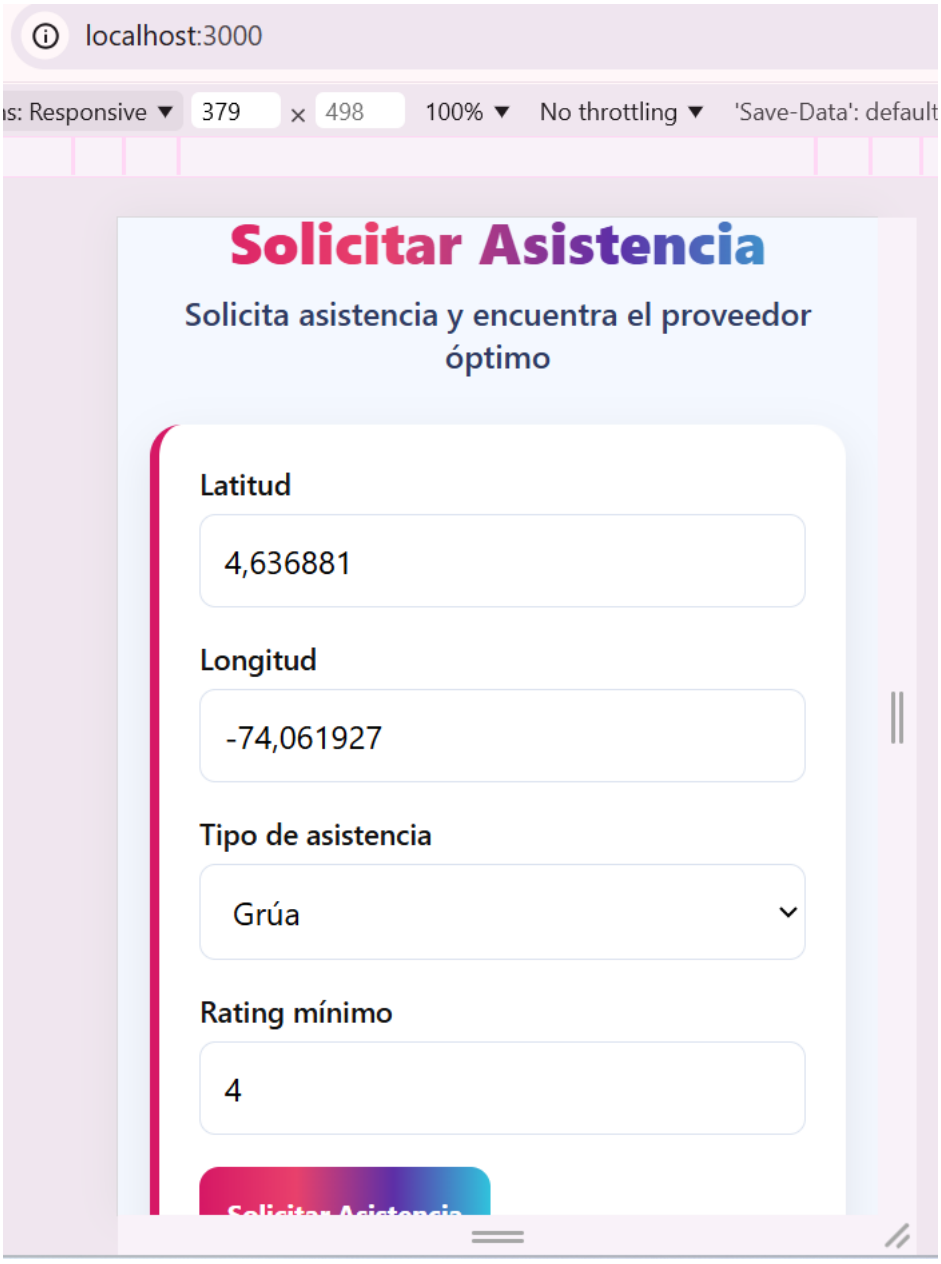
- dotnet test → ejecuta todas las pruebas.
- dotnet test <project> → ejecuta tests por proyecto.
- Integration tests requieren PostgreSQL con Docker.
- Se incluyen pruebas unitarias, de lógica de negocio y de acceso a datos.

Casos de prueba y evidencias

Prueba Navegador WEB



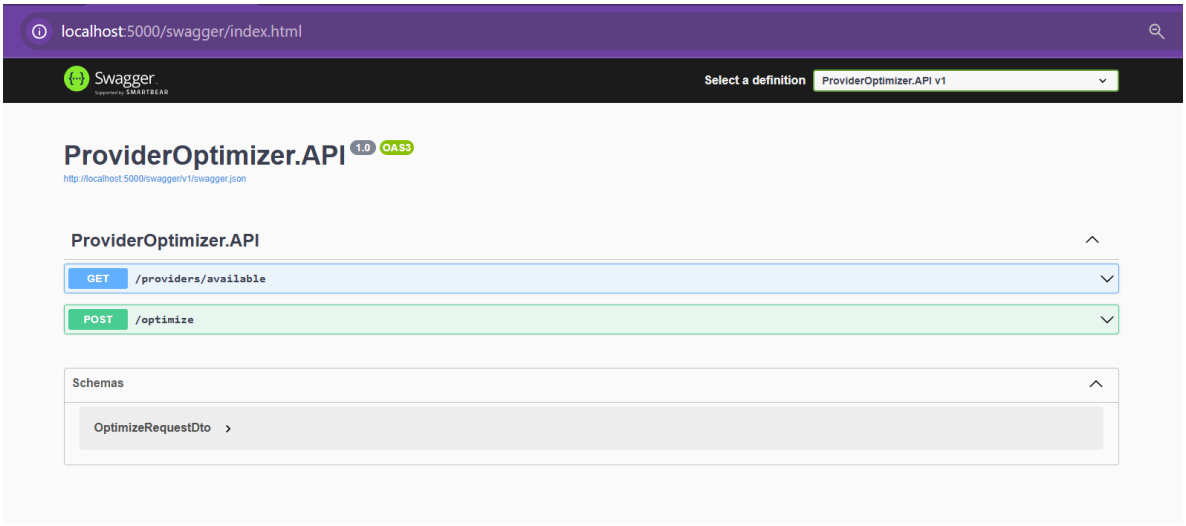
MOVIL

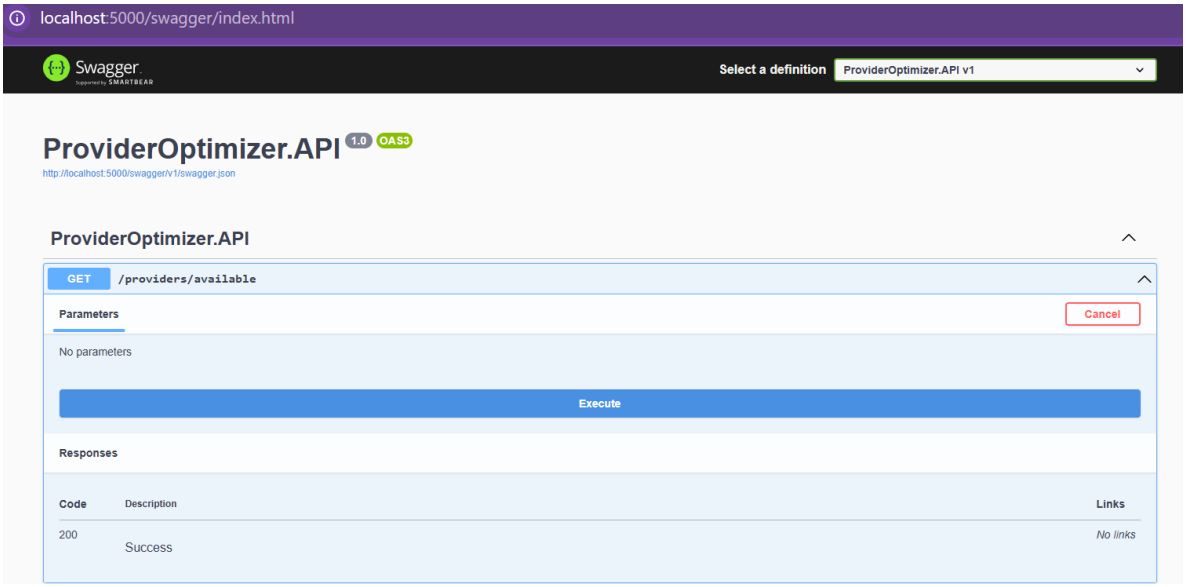

localhost:3000
s: Responsive
379
498
100%
No throttling
'Save-Data': default
Solicitar Asistencia
Solicita asistencia y encuentra el proveedor óptimo
Latitud
4,636881
Longitud
-74,061927
Tipo de asistencia
Grúa
Rating mínimo
4

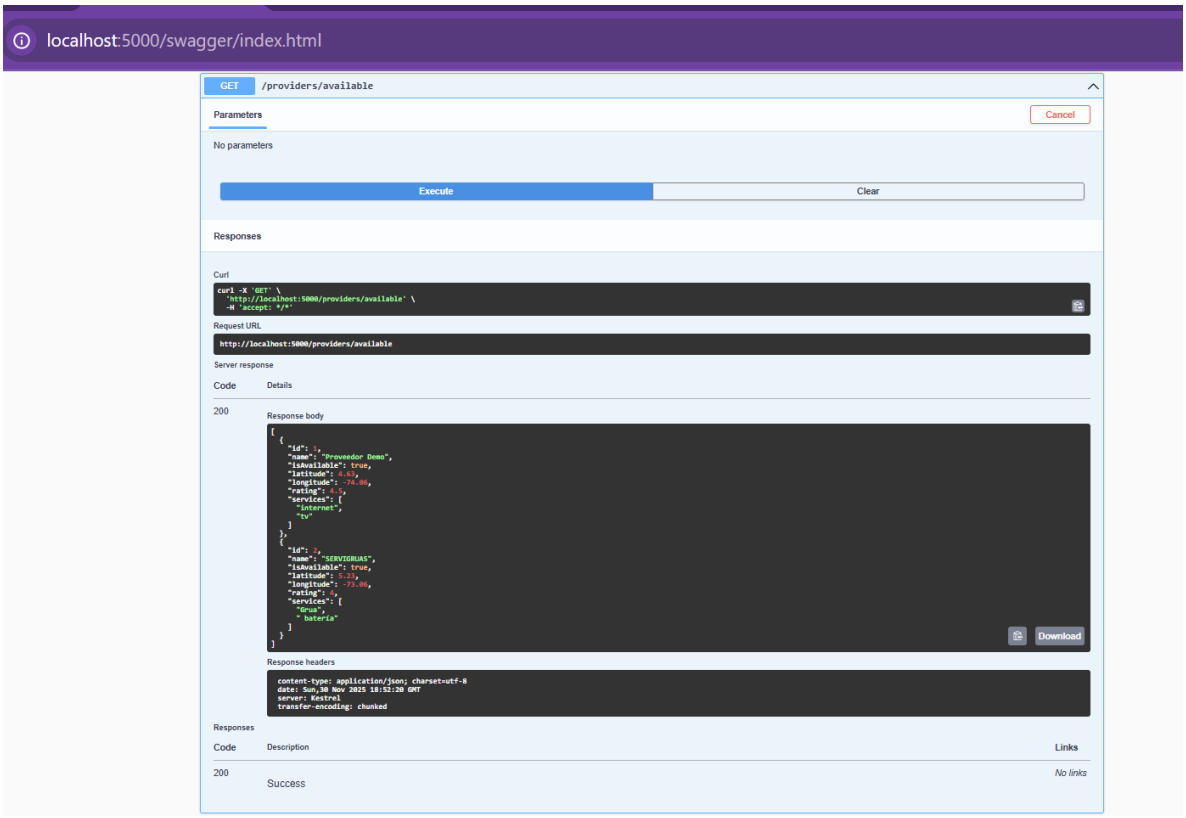
Se verifica el registro en BD

```
provider_optimizer=# select * from optimizations;
 id | requestname | latitude | longitude  | rating | providerid
----+-------------+----------+------------+--------+------------
  1 | grua        |     3423 |       4555 |      6 |          1
  2 | internet    |       54 |         78 |      5 |          1
  3 | bateria     | 4.639949 | -74.062234 |      5 |          1
  4 | cerrajeria  | 4.639938 | -74.062255 |    4.5 |          1
  5 | cerrajeria  | 5.639955 | -84.062234 |      5 |          1
  6 | cerrajeria  | 4.639949 | -74.062234 |      4 |          1
  7 | grua        | 4.636835 | -74.062227 |      4 |          1
  8 | Grua        |      5.2 |      -78.9 |      4 |          1
  9 | grua        | 4.636881 | -74.061927 |      4 |          1
(9 rows)
```

Prueba Swagger





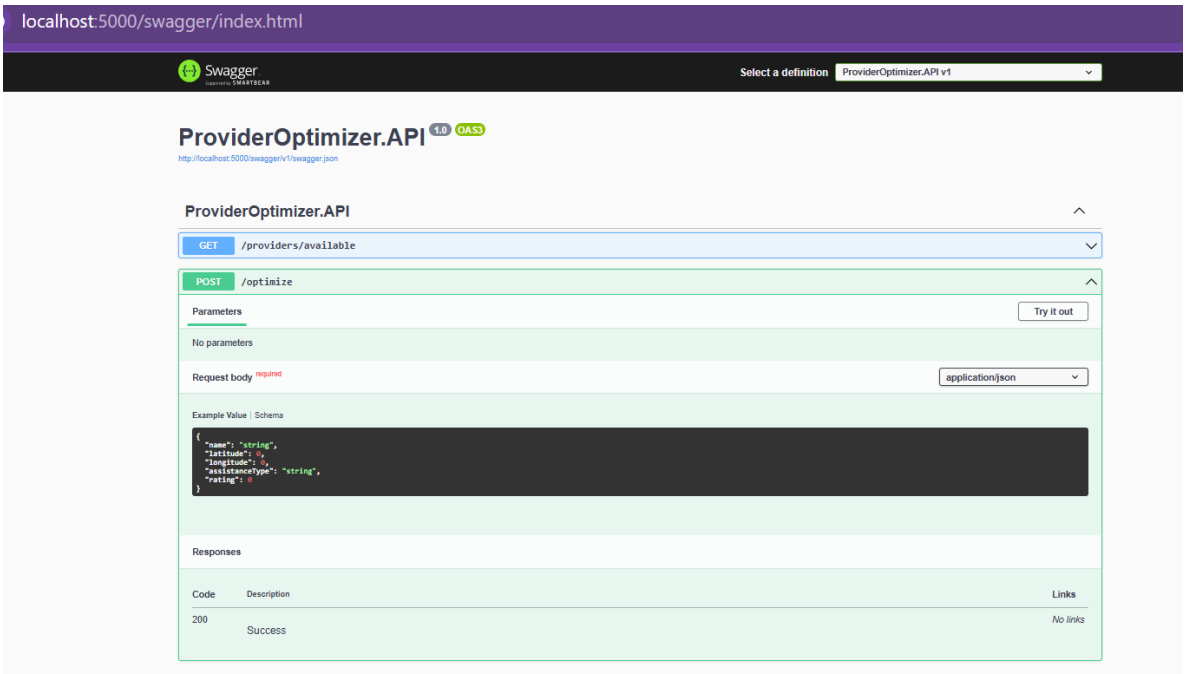


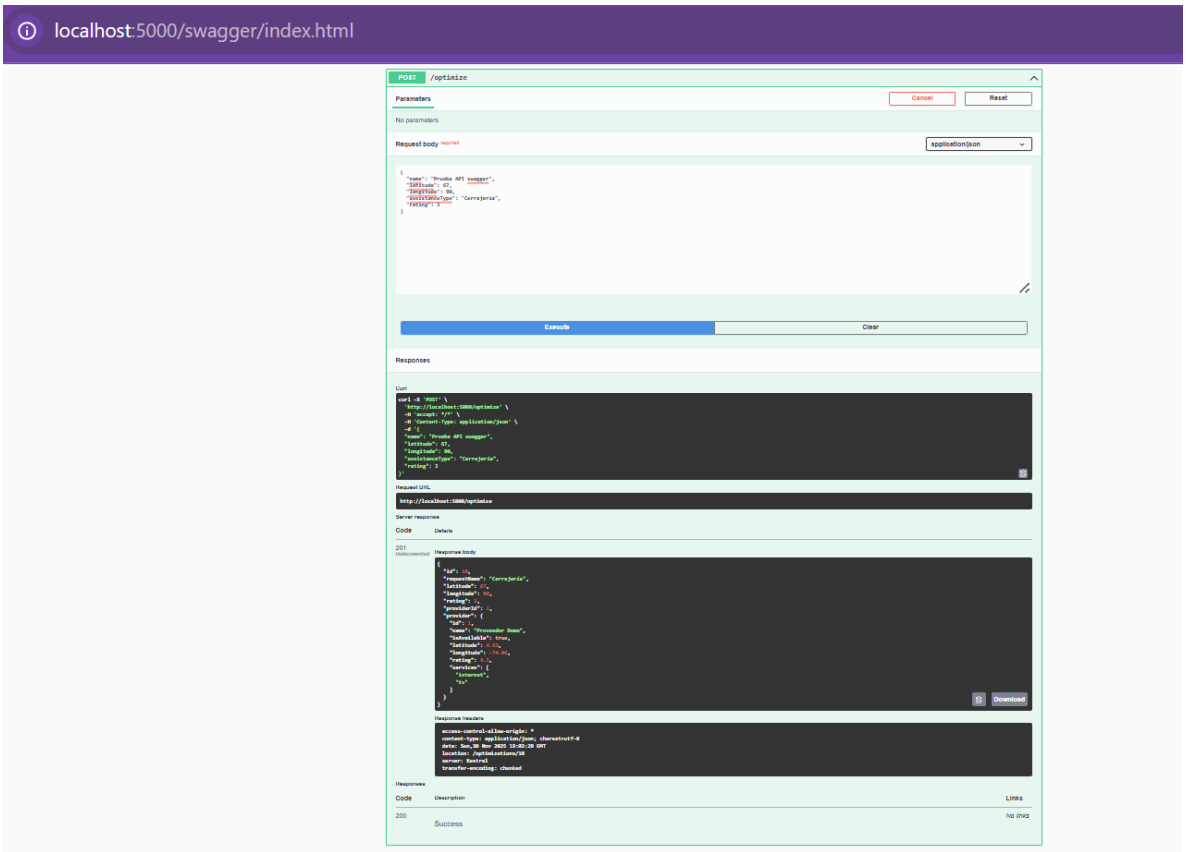

```
curl -X 'GET' \
  'http://localhost:5000/providers/available' \
  -H 'accept: */*'
```

```
[
  {
    "id": 1,
    "name": "Proveedor Demo",
    "isAvailable": true,
    "latitude": 4.63,
    "longitude": -74.06,
    "rating": 4.5,
    "services": [
      "internet",
      "tv"
    ]
```

```
  },
  {
    "id": 2,
    "name": "SERVIGRUAS",
    "isAvailable": true,
    "latitude": 5.23,
    "longitude": -73.06,
    "rating": 4,
    "services": [
      "Grua",
      " batería"
    ]
  }
]
```


```
provider_optimizer=# select * from providers;
 id |      name      | latitude | longitude | isavailable | rating |   services
----+----------------+----------+-----------+-------------+--------+--------------
  1 | Proveedor Demo |     4.63 |    -74.06 | t           |    4.5 | internet,tv
  2 | SERVIGRUAS     |     5.23 |    -73.06 | t           |      4 | Grua, batería
```



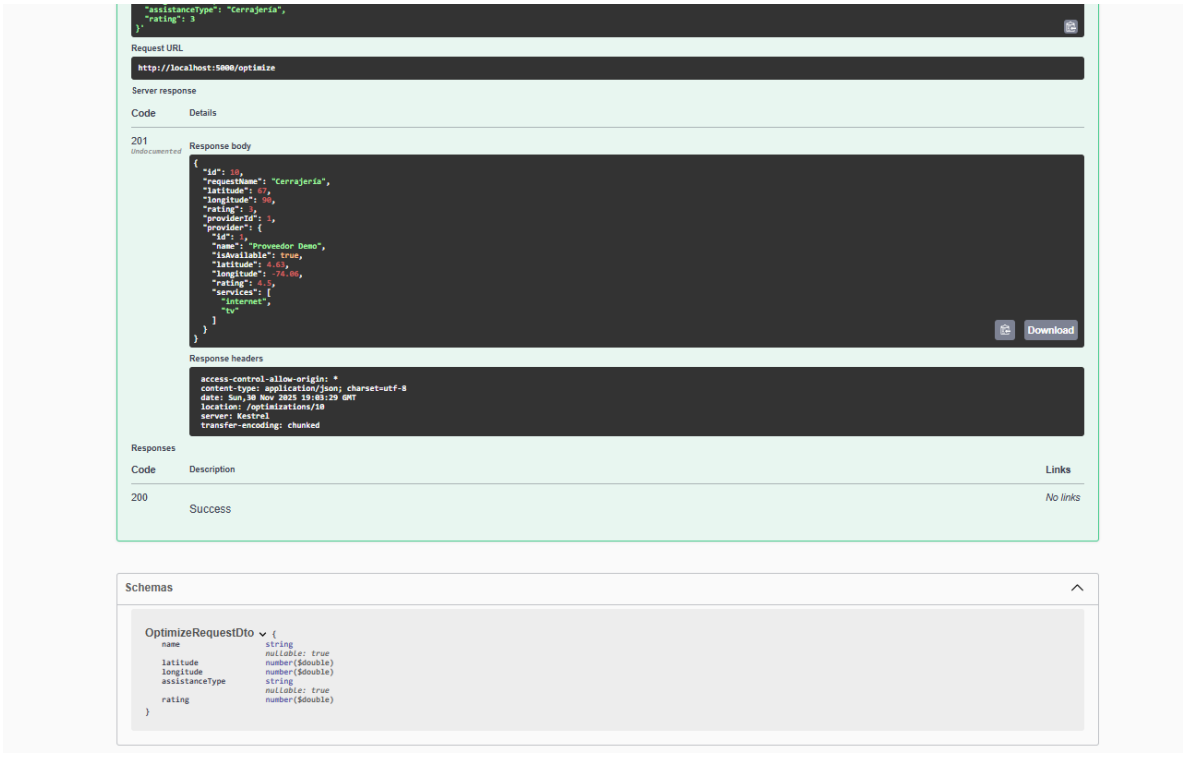



```
curl -X 'POST' \
  'http://localhost:5000/optimize' \
  -H 'accept: */*' \
```

```
  -H 'Content-Type: application/json' \
  -d '{
  "name": "Prueba API swagger",
  "latitude": 67,
  "longitude": 90,
  "assistanceType": "Cerrajería",
  "rating": 3
}'
```

Response


```
{
  "id": 10,
  "requestName": "Cerrajería",
  "latitude": 67,
  "longitude": 90,
  "rating": 3,
  "providerId": 1,
  "provider": {
    "id": 1,
    "name": "Proveedor Demo",
    "isAvailable": true,
    "latitude": 4.63,
    "longitude": -74.06,
    "rating": 4.5,
    "services": [
      "internet",
      "tv"
    ]
  }}
```




```
provider_optimizer=# select * from optimizations;
 id | requestname | latitude | longitude  | rating | providerid
----+-------------+----------+------------+--------+------------
  1 | grua        |     3423 |       4555 |      6 |          1
  2 | internet    |       54 |         78 |      5 |          1
  3 | bateria     | 4.639949 | -74.062234 |      5 |          1
  4 | cerrajeria  | 4.639938 | -74.062255 |    4.5 |          1
  5 | cerrajeria  | 5.639955 | -84.062234 |      5 |          1
  6 | cerrajeria  | 4.639949 | -74.062234 |      4 |          1
  7 | grua        | 4.636835 | -74.062227 |      4 |          1
  8 | Grua        |      5.2 |      -78.9 |      4 |          1
  9 | grua        | 4.636881 | -74.061927 |      4 |          1
 10 | Cerrajería  |       67 |         90 |      3 |          1
(10 rows)
```

5. Conclusiones

- Todos los módulos críticos fueron probados y funcionan de acuerdo con los requerimientos.
- El sistema cumple con los criterios de aceptación definidos en el plan de trabajo.
- Se recomienda avanzar a la fase de despliegue piloto.