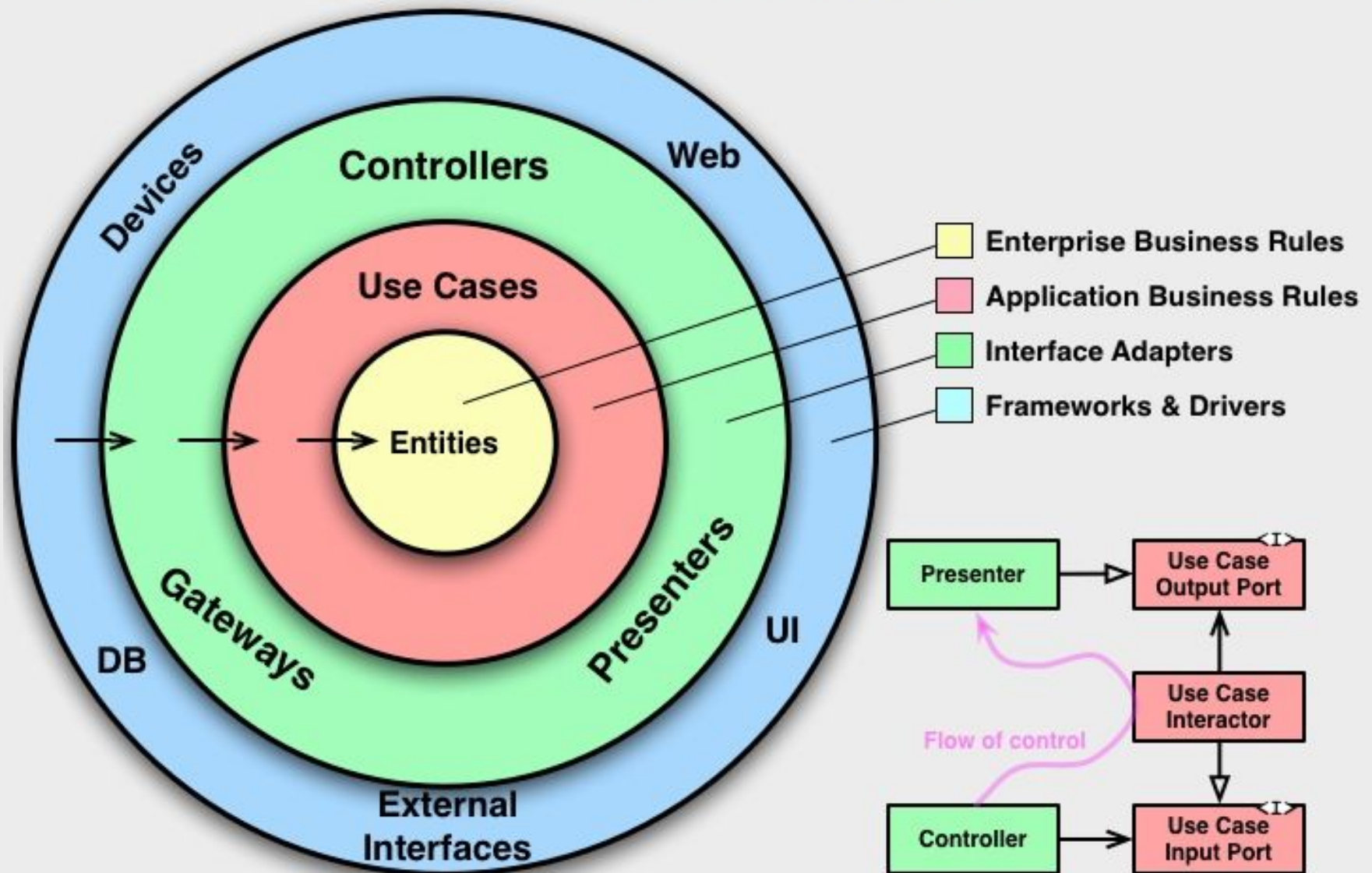


# Curso de NodeJS con PostgreSQL



# The Clean Architecture





# Nicolas Molina

Google Developer Expert  
Dev and Teacher at Platzi  
@nicobytes

Search product

Order: Most recent



\$ 35,00  
Round shelf



\$ 120,00  
Retro refrigerator



\$ 35,00  
Round shelf



\$ 120,00  
Retro refrigerator



\$ 35,00  
Round shelf



# **Corriendo Postgres con Docker**



```
docker-compose up -d postgres
docker-compose ps
docker-compose down
```

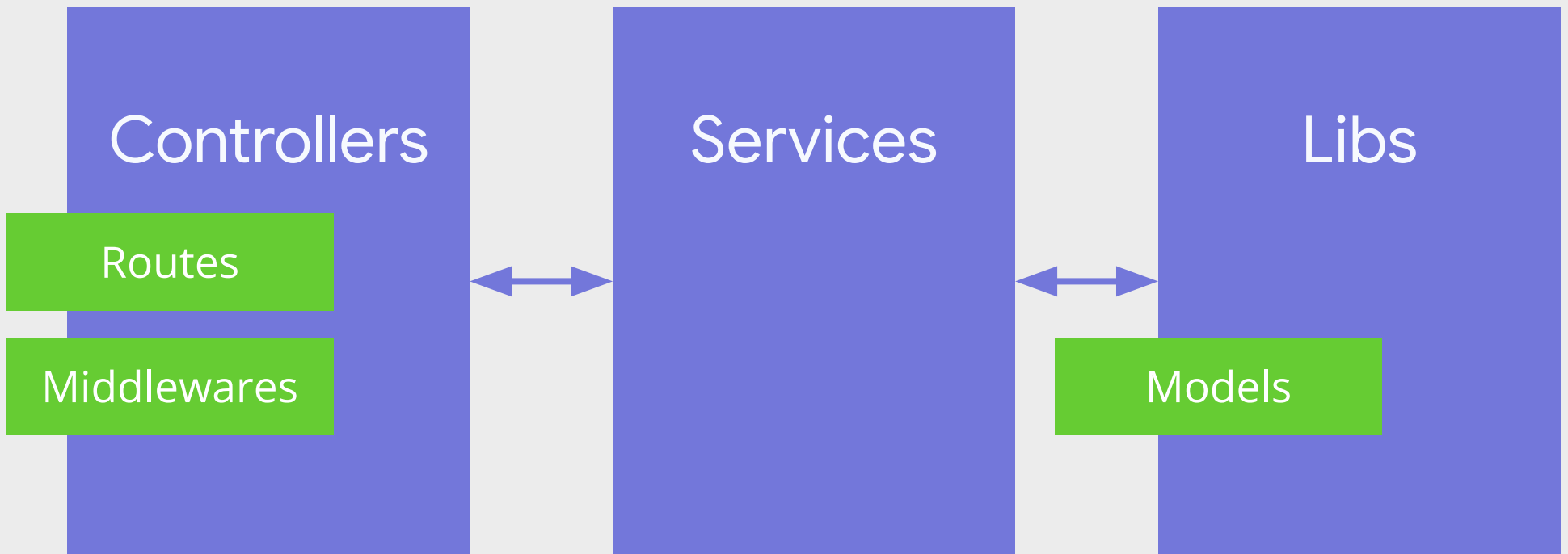
```
docker-compose up -d postgres
docker-compose ps
docker-compose down
```



# **Explorando nuestra DB**

# **Integración con Node Postgres**

# The Clean Architecture



# **Manejando un Pool**

# **Variables de Ambiente**

**¿Qué es un  
ORM?**

# **Tu primer modelo**



# **CRUD desde el ORM**

# Usando MySql

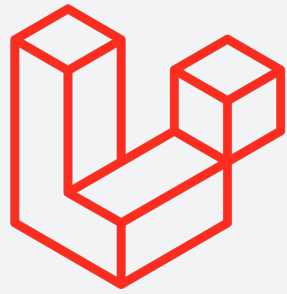
# Migraciones



# django

*Migrations are Django's way of **propagating changes** you make to your models (adding a field, deleting a model, etc.) into your database schema.*





# Laravel

*Migrations are like **version control** for your database, allowing your team to define and share the application's database schema definition.*



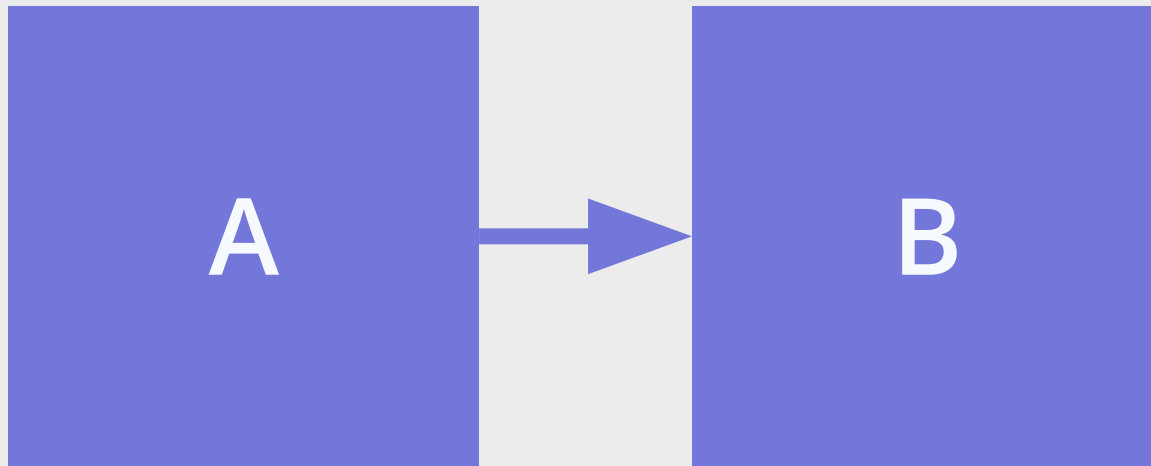
*Just like you use **version control** systems such as Git to manage changes in your source code, you can use migrations to keep track of **changes to the database.***



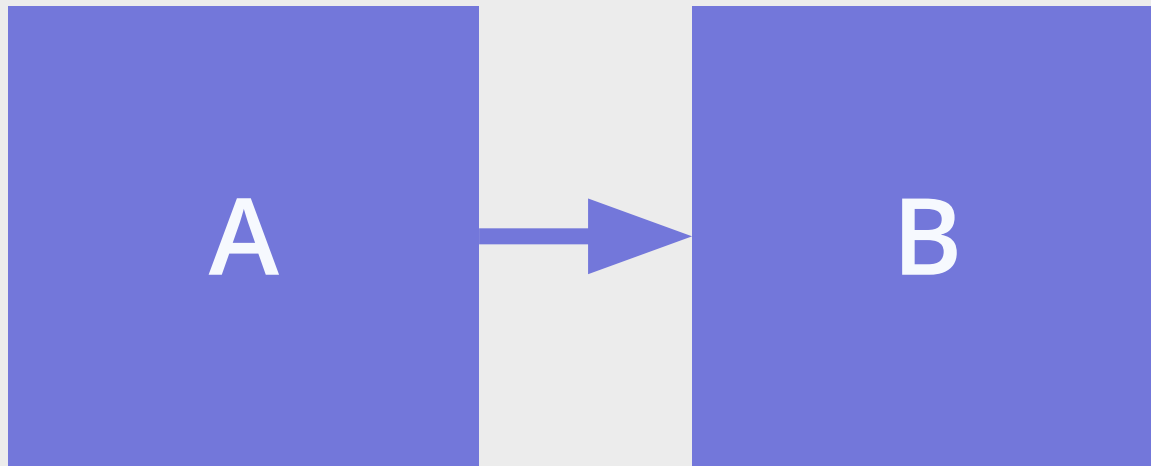
# Migraciones

# Relaciones 1-1

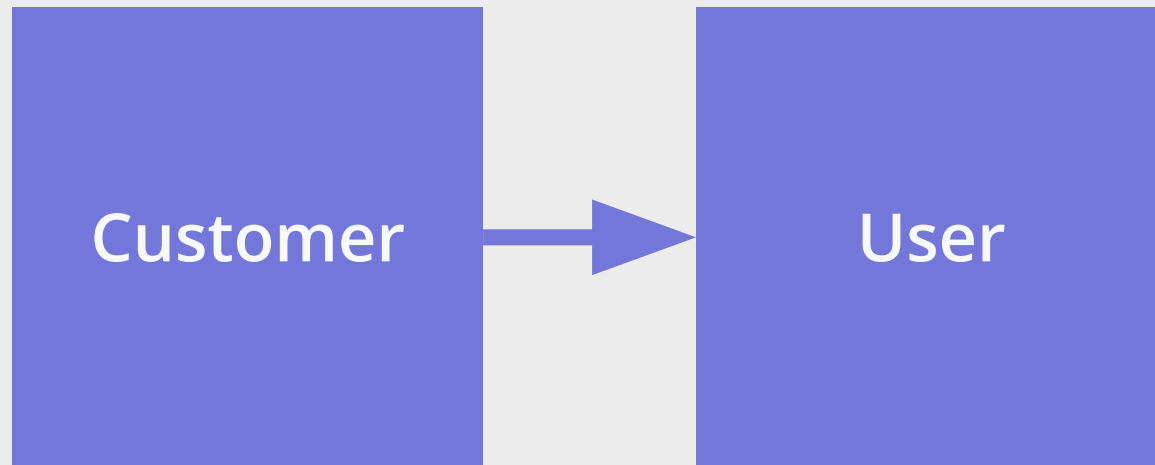
# HasOne -> B



# BelongsTo -> A

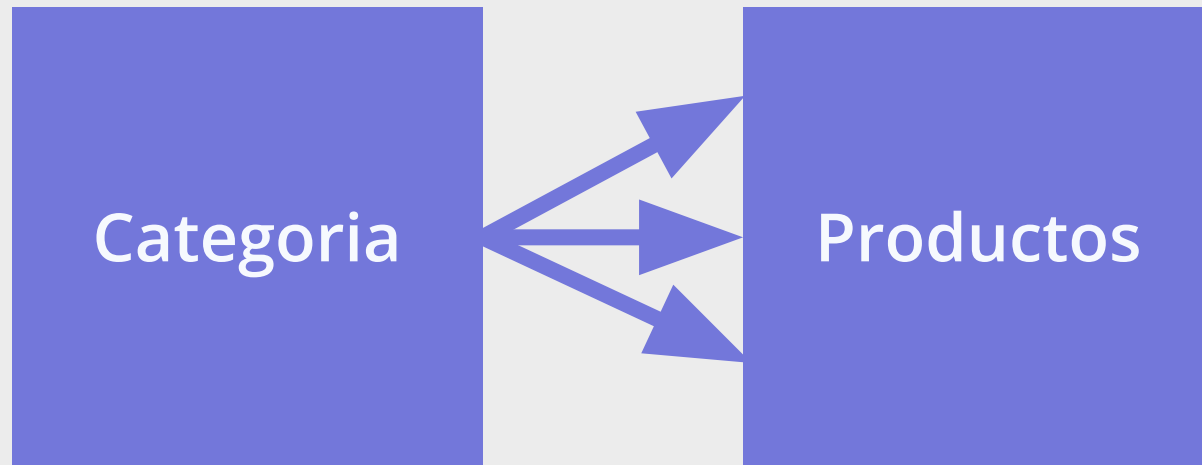


# BelongsTo -> Customer



# Relaciones 1-N

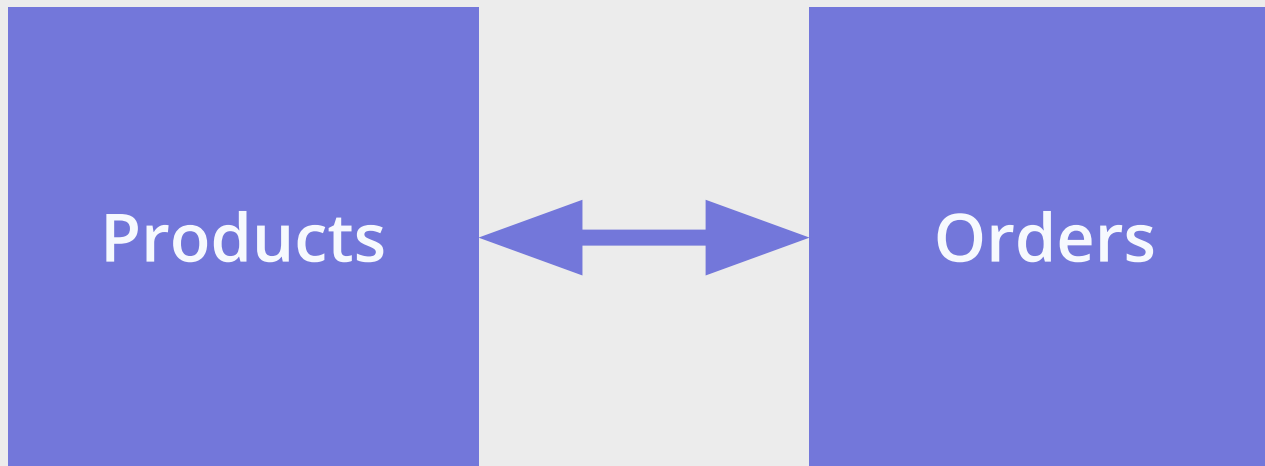
# hasMany -> Products





# Relaciones N-N

# belongsToMany

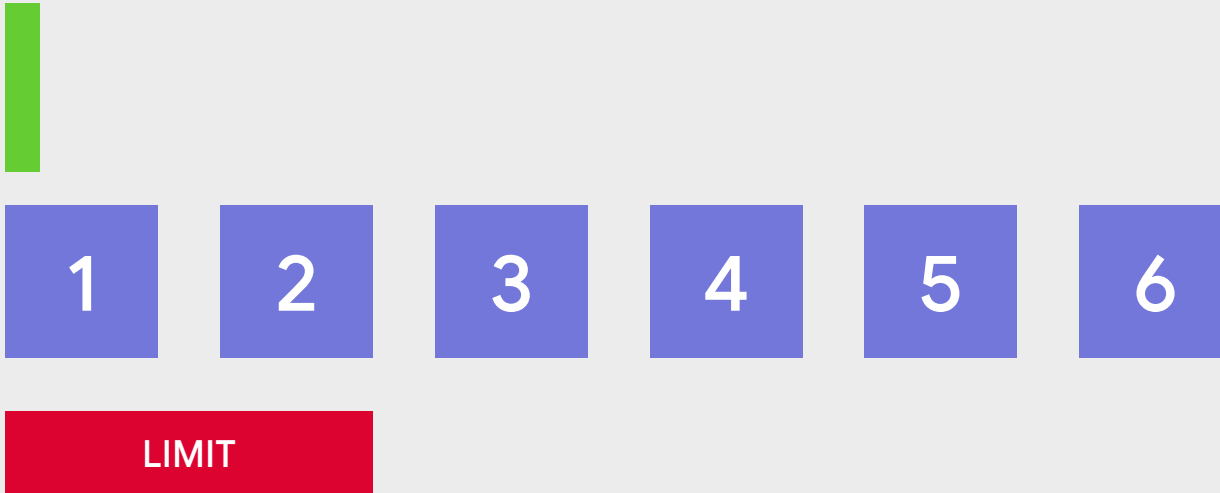


# Paginación

# Paginación con limit & offset

Limit = 2

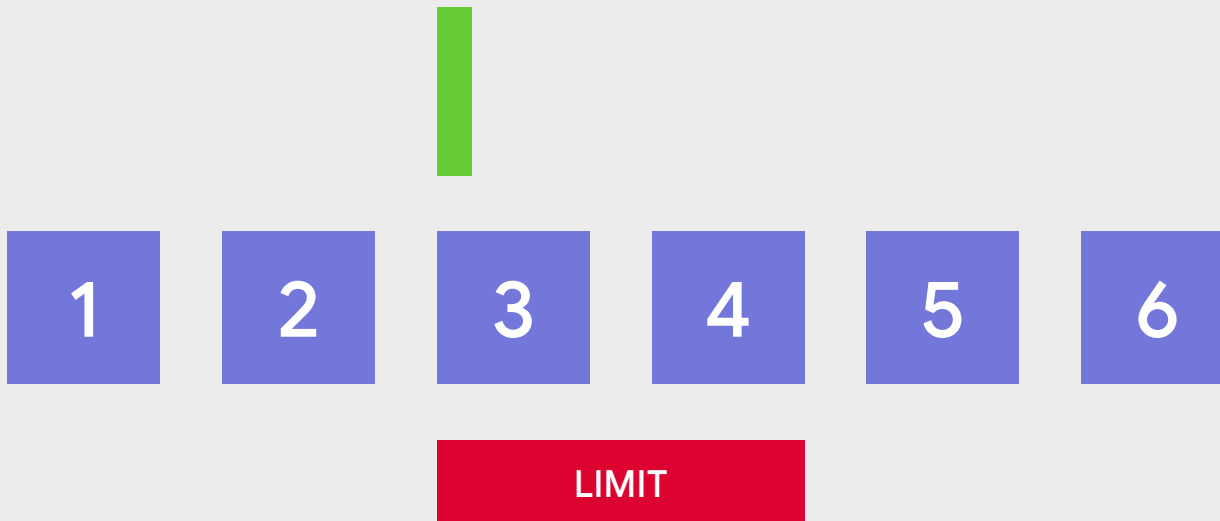
Offset = 0



# Paginación con limit & offset

Limit = 2

Offset = 2



# Paginación con limit & offset

Limit = 2

Offset = 4



**Filtrando  
precios**



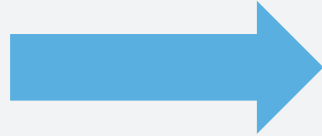
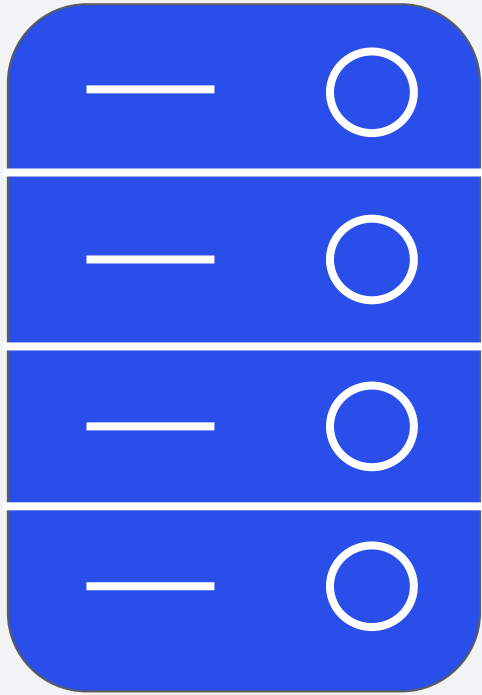
# **Migraciones en Prod**

# Routing

# API Restful

# REST

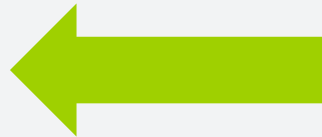
Representational State Transfer



**GET**



**PUT**





**POST**



**DELETE**



Method	/products	/products/{id}
GET	Get list	Get
PUT	Replace 	Update/Replace
PATCH	No Apply	Update
POST	Create	No Apply
DELETE	Delete 	Delete

**GET**

api.example.com/tasks/{id}/  
api.example.com/people/{id}/  
api.example.com/users/{id}/tasks/



# Query Params

api.example.com/products

api.example.com/products?page=1

api.example.com/products?limit=10&offset=0

api.example.com/products?region=USA

api.example.com/products?region=USA&brand=XYZ

# **Single Responsibility Principle**



**POST**

api.example.com/api/v1/users/

api.example.com/api/v1/tasks/

api.example.com/api/v1/customers/

**DELETE**  
**PUT**  
**PATCH**

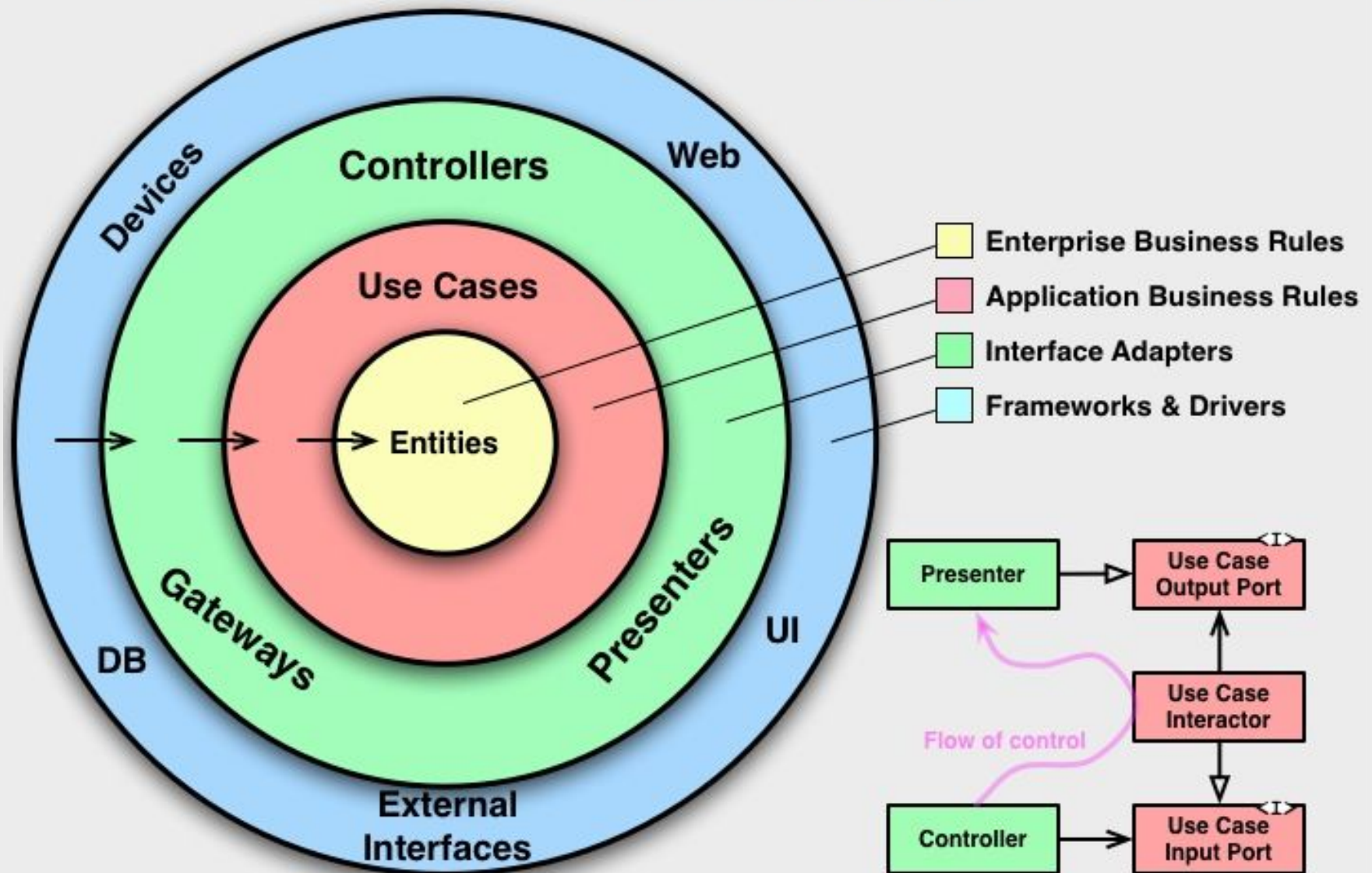
Method	/products	/products/{id}
GET	Get list	Get
PUT	Replace 	Update/Replace
PATCH	No Apply	Update
POST	Create	No Apply
DELETE	Delete 	Delete

**STATUS CODE**

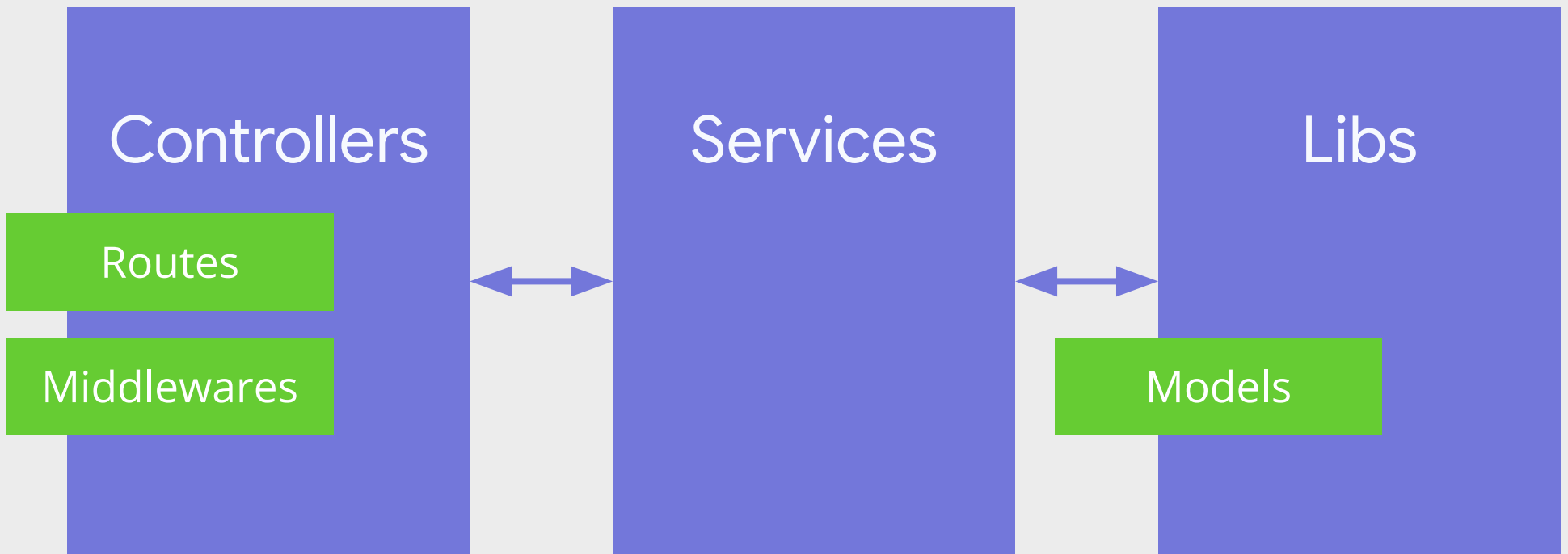


# **SERVICIOS**

# The Clean Architecture



# The Clean Architecture



**Create,  
update & delete**

# Async

# Middlewares





```
graph LR; A[Middleware] --> B[Middleware]; B --> C[Middleware];
```

Middleware

Middleware

Middleware



```
graph LR; M1[Middleware] --> M2[Middleware]; M3[Middleware];
```

Middleware

Middleware

Middleware

```
function (req, res, next) {  
  if (something) {  
    res.send( 'end' );  
  } else {  
    next();  
  }  
}
```

```
function (error, req, res, next) {  
  if (error) {  
    res.status(500).json({error});  
  } else {  
    next();  
  }  
}
```

# Uses Cases

- Funcionan como pipes.
- Validar datos.
- Capturar errores.
- Validar permisos.
- Controlar accesos.

```
{  
  "user": {  
    "id": 4,  
    "email": "superadmin@mail.com",  
    "role": "admin",  
    "createdAt": "2021-08-31T16:26:06"  
  },  
  "token": "...."  
}
```

# Middleware for Errors

# **Manejo de errores con Boom**

# **Validando el Datos**



# **Probando los endpoints**

# **Consideraciones para prod**

# Recomendaciones

- Cors
- Https
- Procesos de Build
- Remover logs
- Seguridad (Helmet)
- Testing

api.mydomain.com



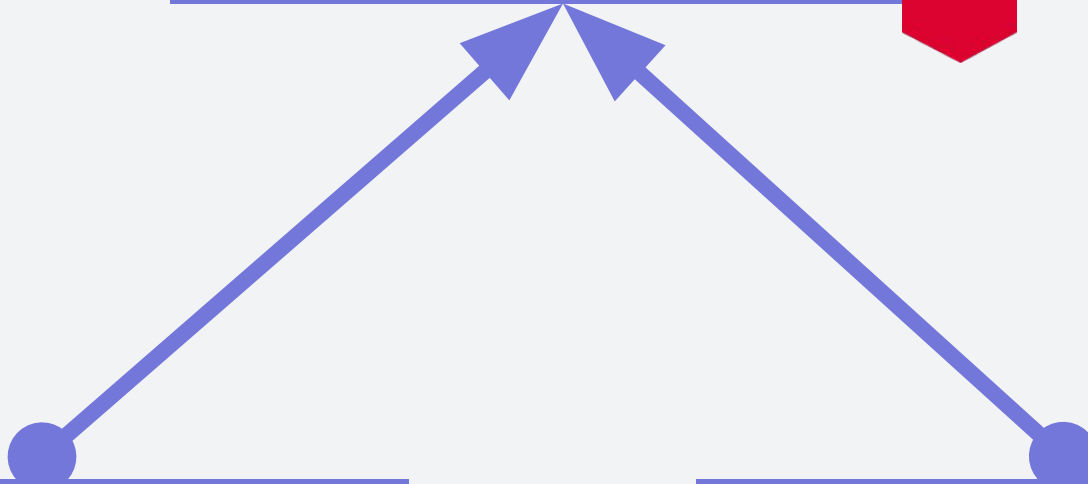
api.mydomain.com

api.mydomain.com



mydomain.com

otro.mydomain.com



**CORS**

# Deployment to Heroku

```
curl https://cli-  
assets.heroku.com/install.sh | sh
```

```
heroku login  
heroku create  
git remote -v
```



# Deployment to Heroku

heroku logs --tail

```
const express = require('express');  
const app = express();  
const port = 3000;
```

```
app.get('/', (req, res) => {  
  res.send('Hello World!')  
});
```

```
app.listen(port, () => {  
  console.log(`http://localhost:${port}`)  
});
```