

# Registro Unificado de Actividades

---

Franco Ganga

Universidad Nacional Arturo Jauretche - Instituto de Ingeniería y Agronomía

La Universidad no posee un sistema que administre la información sobre las actividades realizadas por cada persona: actualmente, estos datos son almacenados en planillas excel o documentos escritos. El propósito de este proyecto es ofrecer una solución a este problema.

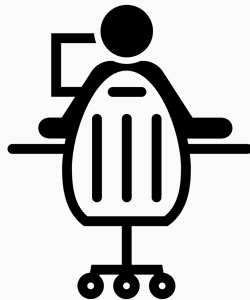
# Objetivos



## Objetivos por alcanzar

---

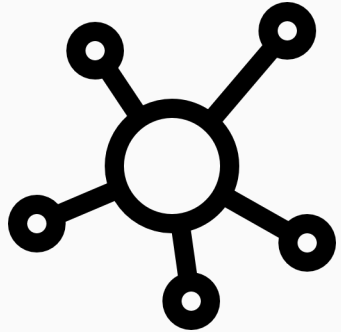
- Plataforma web de administración.
- Integración con datos de sistemas externos.
- Servicio API REST.



## Objetivos por alcanzar

---

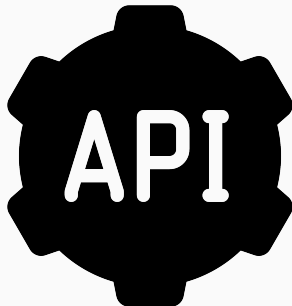
- Plataforma web de administración.
- Integración con datos de sistemas externos.
- Servicio API REST.



## Objetivos por alcanzar

---

- Plataforma web de administración.
- Integración con datos de sistemas externos.
- Servicio API REST.



- **Symfony**
- Doctrine
- Sonata
- Api-Platform



## Componentes utilizados

- Symfony
- Doctrine
- Sonata
- Api-Platform





## Componentes utilizados

- Symfony
- Doctrine
- Sonata
- Api-Platform



## Componentes utilizados

- Symfony
- Doctrine
- Sonata
- Api-Platform



## Características

---



- Es user-friendly.
- Herramientas CLI.
- Fácil de configurar.
- Comunidad grande.
- Buena documentación

## Características

---



- Es user-friendly.
- Herramientas CLI.
- Fácil de configurar.
- Comunidad grande.
- Buena documentación

## Características

---



- Es user-friendly.
- Herramientas CLI.
- Fácil de configurar.
- Comunidad grande.
- Buena documentación

## Características

---



- Es user-friendly.
- Herramientas CLI.
- Fácil de configurar.
- Comunidad grande.
- Buena documentación

## Características

---



- Es user-friendly.
- Herramientas CLI.
- Fácil de configurar.
- Comunidad grande.
- Buena documentación

# Sonata





## Qué es Sonata

---

- Interfaz de administración
- Integración con Doctrine
- Integración con FOSUser



## Qué es Sonata

---

- Interfaz de administración
- Integración con Doctrine
- Integración con FOSUser




## Qué es Sonata


---


- Interfaz de administración
- Integración con Doctrine
- Integración con FOSUser




# Acciones: Lista
















 Admin Unaj

Buscar 


admin 


- » Asambleaista
- » Consejero Superior
- » Director de Instituto
- » Persona
- » Proyecto de Investigación
- » Carrera
- » Director de Carrera
- » Coordinador de Materia
- » Proyecto de Extensión
- » Rol de Proyecto
- » Comisión de Consejo Superior


 / Proyecto Investigación List

<input type="checkbox"/>	Nombre	Action
<input type="checkbox"/>	et	 Mostrar  Editar  Borrar
<input type="checkbox"/>	non	 Mostrar  Editar  Borrar
<input type="checkbox"/>	nemo	 Mostrar  Editar  Borrar
<input type="checkbox"/>	magnam	 Mostrar  Editar  Borrar
<input type="checkbox"/>	facilis	 Mostrar  Editar  Borrar
<input type="checkbox"/>	Todos los elementos (5)	<div><input type="button" value="Borrar"/> <input type="button" value="OK"/></div>

# Acciones: Creación

Admin Unaj

Buscar 

admin 

- » Asambleaísta
- » Consejero Superior
- » Director de Instituto
- » Persona
- » Proyecto de Investigación
- » Carrera
- » Director de Carrera
- » Coordinador de Materia
- » Proyecto de Extensión
- » Rol de Proyecto
- » Comisión de Consejo Superior

 /  / Proyecto Investigacion List / Proyecto Investigacion Create

Crear

### Proyecto de Investigación

**Nombre \***

**Roles \***


 Agregar nuevo

 Crear y editar

 Crear y regresar al listado

 Crear y agregar otro


# Acciones: Edición

 Admin Unaj

Buscar

admin

- » Asambleaísta
- » Consejero Superior
- » Director de Instituto
- » Persona
- » Proyecto de Investigación
- » Carrera
- » Director de Carrera
- » Coordinador de Materia
- » Proyecto de Extensión
- » Rol de Proyecto

 / Proyecto Investigacion List / et

Editar "et" Ver Proyecto Editar Proyecto Administrar Miembros

Proyecto de Investigación

**Nombre \***

et

**Roles \***

Agregar nuevo

Actualizar Actualizar y cerrar Borrar

## ORM: Doctrine

---

- Basado en Hibernate
- Integración con Sonata
- Migraciones



## ORM: Doctrine

---

- Basado en Hibernate
- Integración con Sonata
- Migraciones





## ORM: Doctrine

---

- Basado en Hibernate
- Integración con Sonata
- Migraciones



# Sonata-User

### Qué es Sonata-User

---



- Integra **Sonata** y **FOSUser**
- Menú de usuario
- Soporte para autenticación mediante Google
- Administración de usuarios y grupos
- Roles

### Qué es Sonata-User

---



- Integra **Sonata** y **FOSUser**
- Menú de usuario
- Soporte para autenticación mediante Google
- Administración de usuarios y grupos
- Roles

### Qué es Sonata-User

---



- Integra **Sonata** y **FOSUser**
- Menú de usuario
- Soporte para autenticación mediante Google
- Administración de usuarios y grupos
- Roles

### Qué es Sonata-User

---



- Integra **Sonata** y **FOSUser**
- Menú de usuario
- Soporte para autenticación mediante Google
- Administración de usuarios y grupos
- Roles

### Qué es Sonata-User

---



- Integra **Sonata** y **FOSUser**
- Menú de usuario
- Soporte para autenticación mediante Google
- Administración de usuarios y grupos
- Roles

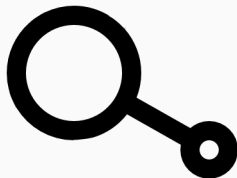
# **Instalación y Configuración**



## Involucra

---

- Doctrine
- FOSUser
- Security
- Routing
- Sonata-User



## Involucra

---

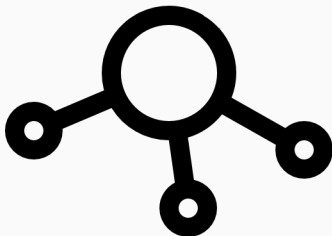
- Doctrine
- FOSUser
- Security
- Routing
- Sonata-User



## Involucra

---

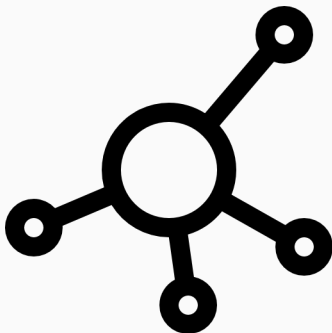
- Doctrine
- FOSUser
- Security
- Routing
- Sonata-User



## Involucra

---

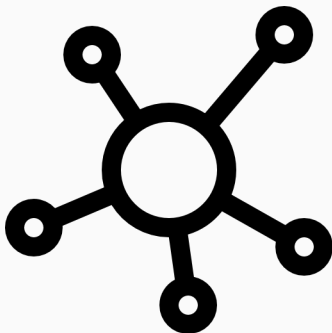
- Doctrine
- FOSUser
- Security
- Routing
- Sonata-User



## Involucra

---

- Doctrine
- FOSUser
- Security
- Routing
- Sonata-User



# API-Platform

## Qué es API Platform

---

- *API framework*
- Soporta muchos estándares de desarrollo



## Qué es API Platform

---

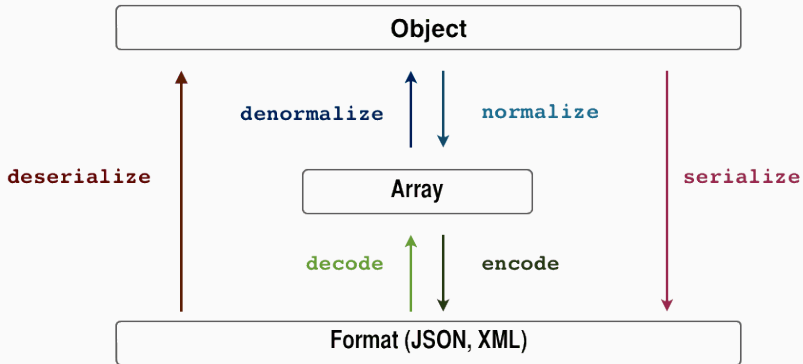
- *API framework*
- Soporta muchos estándares de desarrollo





# Serialización

# Serialización



# **Contexto de Normalización y Desnormalización**

Definen propiedades presentes en  
cada uno de los procesos

## Grupos

---

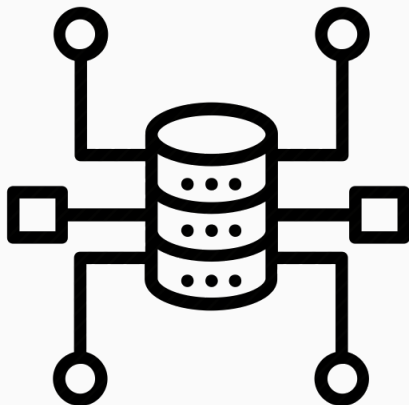
- Permiten agrupar propiedades en cada contexto
- Siguen relaciones

## Grupos

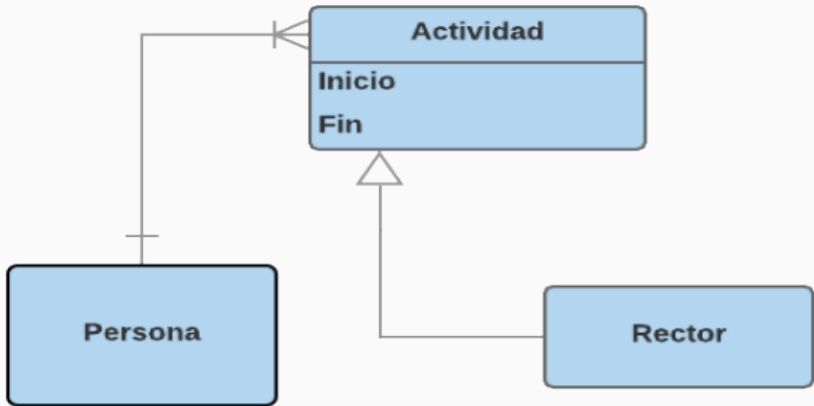
---

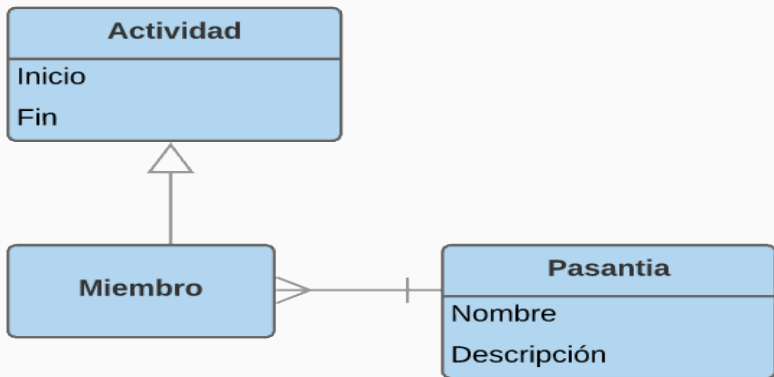
- Permiten agrupar propiedades en cada contexto
- Siguen relaciones

# Modelado

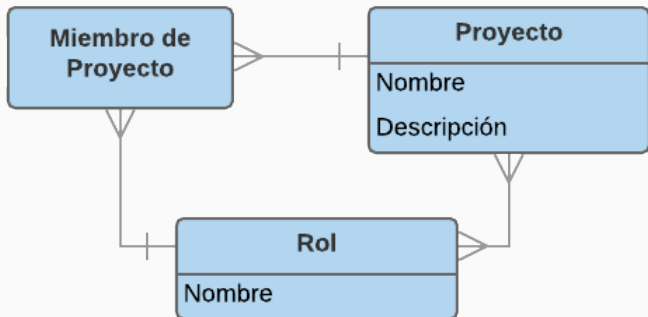


# Actividad









# Herencia de Clase

Cada tabla *hija* almacena sus datos  
en una tabla propia

Las tablas *hijas* poseen una referencia  
a la id del *padre*

## Configuración

---

- Propiedad **discriminator**

## Configuración

---

- Propiedad **discriminator**
- Establecer la Herencia de Clase

```
{"miembro_cursoExtension" = "MiembroCursoExtension",  
"rector" = "Rector"}
```

# Herencia de Clase

```
{"miembro_cursoExtension" = "MiembroCursoExtension",  
"rector" = "Rector"}
```

```
class MiembroCursoExtension extends Actividad
```



# **Mapeo Objeto-Relacional**

## Mapeo Objeto - Relacional: Metadata (Anotaciones)

```
<?php
/** @Entity */
class Message
{
    /** @Column(type="integer") */
    private $id;
    /** @Column(length=140) */
    private $text;
    /** @Column(type="datetime", name="posted_at") */
    private $postedAt;
}
```

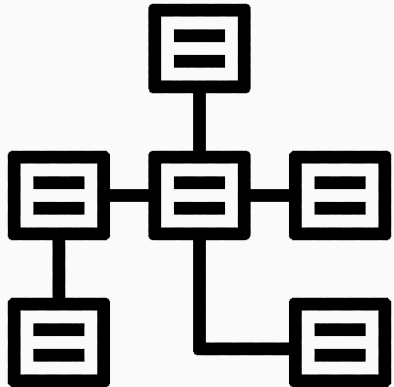
## Mapeo Objeto - Relacional: Metadata (XML)

```
<doctrine-mapping>
  <entity name="Message">
    <field name="id" type="integer" />
    <field name="text" length="140" />
    <field name="postedAt" column="posted_at" type="datetime" />
  </entity>
</doctrine-mapping>
```

## Mapeo Objeto - Relacional: Metadata (YAML)

```
Message:
  type: entity
  fields:
    id:
      type: integer
    text:
      length: 140
    postedAt:
      type: datetime
      column: posted_at
```

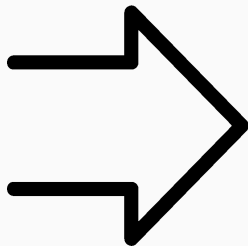
# Relaciones



## Tipos de Relaciones

---

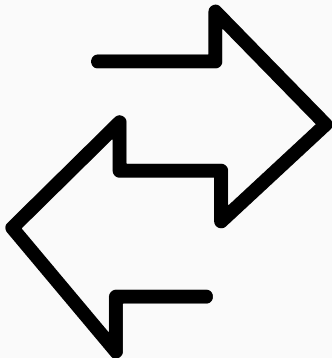
- Unidireccionales
- Bidireccionales



## Tipos de Relaciones

---

- Unidireccionales
- Bidireccionales



# **Lado Propietario**



## **Lado Propietario**

Doctrine solo comprueba este lado  
por cambios en la relación

# **Clases Administradoras**

Definen qué hacer en cada acción

## Qué es necesario

---

- Definir la Clase
- Establecerla como servicio



## Qué es necesario

---

- Definir la Clase
- Establecerla como servicio



# Ejemplos



```
public function configureFormFields(  
    FormMapper $formMapper  
): void  
{  
    $formMapper  
        ->add('persona', ModelListType::class)  
        ->add('inicio', DatePickerType::class)  
        ->add('fin', DatePickerType::class)  
        ;  
}
```

```
public function configureListFields(  
    ListMapper $listMapper  
): void  
{  
    $listMapper  
        ->add('persona.nombre', null, [  
            'label' => 'Nombre'  
        ])  
        ->add('persona.apellido', null, [  
            'label' => 'Apellido'  
        ])  
        ->add('inicio')  
        ->add('fin')  
}
```

```
public function configureShowFields(  
    ShowMapper $showMapper  
): void  
{  
    $showMapper  
        ->add('persona.nombre', null, [  
            'label' => 'Nombre'  
        ])  
        ->add('persona.apellido', null, [  
            'label' => 'Apellido'  
        ])  
        ->add('inicio')  
        ->add('fin')  
        ;  
}
```



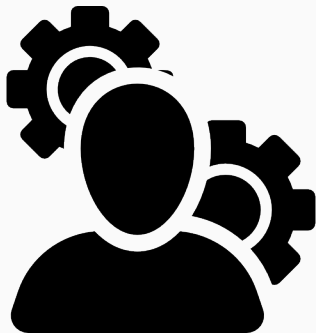
```
admin.movilidad_rtf:  
  class: App\Admin\MovilidadRTFAdmin  
  arguments: [~, App\Entity\MovilidadRTF, ~]  
  tags:  
    - {  
      name: sonata.admin,  
      manager_type: orm,  
      group: admin,  
      label: MovilidadRTF  
    }  
  public: true
```

# Admins

### Actividades simples

---

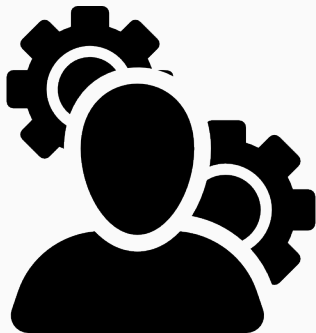
- Involucran una única persona
- Duplicación de código



## Actividades simples

---

- Involucran una única persona
- Duplicación de código



## Traits

---



- Métodos redefinidos por orden de precedencia
- Cada admin mantiene su propia clase
- Más flexible

## Traits

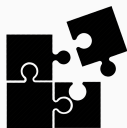
---



- Métodos redefinidos por orden de precedencia
- Cada admin mantiene su propia clase
- Más flexible

## Traits

---



- Métodos redefinidos por orden de precedencia
- Cada admin mantiene su propia clase
- Más flexible

# **Admins Compuestos**



En qué consiste

---



### En qué consiste

---



- Cada acción del *hijo* es accedida a través del *padre*

### En qué consiste

---



- Cada acción del *hijo* es accedida a través del *padre*
- Se obtienen rutas anidadas

### Rutas base de Miembros de proyecto:

`/miembroproyecto/id/(show | edit)`  
`/miembroproyecto/list`



Resultado de definir Proyecto  
como padre de miembro:

`/proyecto/{id}/miembroproyecto/{id}/(show | edit)`

`/proyecto/{id}/miembroproyecto/list`

- `[addChild, ['@admin.miembro_proyecto', 'proyecto']]`

# Resultado

  / Proyecto Investigacion List / et / Miembro Proyecto List

Editar "et"    Ver Proyecto    Editar Proyecto    Administrar Miembros



<input type="checkbox"/>	Nombre	Apellido
<input type="checkbox"/>	Carlos Héctor Daniel	ALLIERA
<input type="checkbox"/>	MARTIN	AMIGUCCI

☐ **Todos los elementos (2)**

Borrar ▼


OK

# Resultado

  / Proyecto Investigacion List / et / Miembro Proyecto List

Editar "et"   Ver Proyecto   Editar Proyecto   **Administrar Miembros**

<input type="checkbox"/>	Nombre	Apellido
<input type="checkbox"/>	Carlos Héctor Daniel	ALLIERA
<input type="checkbox"/>	MARTIN	AMIGUCCI

☐ **Todos los elementos (2)**  



# Vista de Persona

### Información a Exponer

---

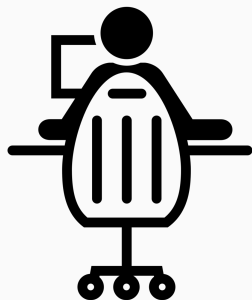
- Listado de Actividades
- Datos de la Persona



### Información a Exponer

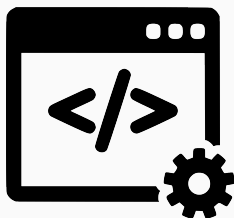
---

- Listado de Actividades
- Datos de la Persona



### Cómo de logró

---

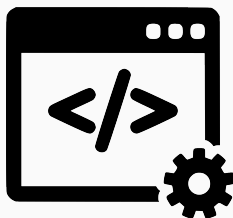


- Iteración de colección de actividades
- Se crea un contenedor por cada actividad
- Se agregan botones y títulos correspondientes

### Cómo de logró

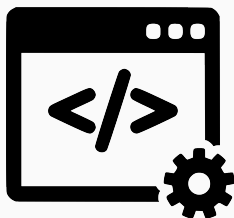
---

- Iteración de colección de actividades
- Se crea un contenedor por cada actividad
- Se agregan botones y títulos correspondientes



### Cómo de logró

---



- Iteración de colección de actividades
- Se crea un contenedor por cada actividad
- Se agregan botones y títulos correspondientes

## Actividades



Miembro de Proyecto

Ocultar



Editar



Ver

**Proyecto:** consequatur

**Rol:** Investigador

**Inicio:** 2018 - Oct - 26

**Fin:** 2021 - Oct - 31

**Estado:** Activa



Miembro de Proyecto

Mostrar



Editar



Ver



Coordinador de Materia

Ocultar



Editar



Ver

**Materia:** et

**Inicio:** 2019 - Oct - 26

**Fin:** 2020 - Jul - 02

**Estado:** Activa



Director de Carrera

Mostrar



Editar



Ver

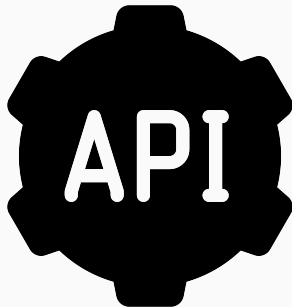
# Integración con Mapuche



### En qué consiste

---

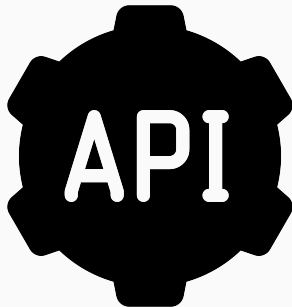
- Obtención de datos externos a través de HTTP
- Llenar entidades con datos obtenidos



### En qué consiste

---

- Obtención de datos externos a través de HTTP
- Llenar entidades con datos obtenidos



# Eventos

## Cómo funcionan

---

- Notificaciones
- Listeners y Subscribers



## Cómo funcionan

---

- Notificaciones
- Listeners y Subscribers



# **Eventos: Doctrine**

Relacionados al ciclo de vida de una entidad

## **PostLoad**

Una vez que la entidad es cargada