

GameSide

Implementa un proyecto web Django para **venta online de videojuegos**.

1. Puesta en marcha

Lleva a cabo los siguientes comandos para la puesta en marcha del proyecto:

```
just create-venv
source .venv/bin/activate
just setup
```

¿Qué ha ocurrido?

- Se ha creado un entorno virtual en la carpeta `.venv`
- Se han instalado las dependencias del proyecto.
- Se ha creado un proyecto Django en la carpeta `main`
- Se han aplicado las migraciones iniciales del proyecto.
- Se ha creado un superusuario con credenciales: `admin - admin`

2. Aplicaciones

Habrás que añadir las siguientes aplicaciones:

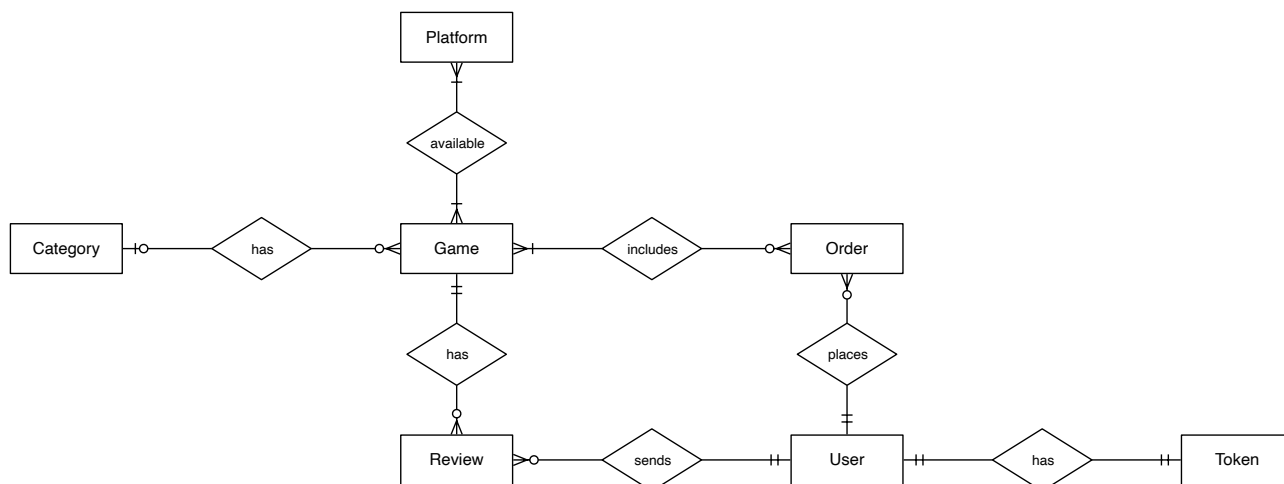
<code>shared</code>	Artefactos compartidos.
<code>games</code>	Gestión de juegos.
<code>platforms</code>	Gestión de plataformas.
<code>categories</code>	Gestión de categorías.
<code>orders</code>	Gestión de pedidos.
<code>users</code>	Gestión de usuarios.

Se proporciona una *receta* `just` para añadir una aplicación:

```
just startapp <app>
```

Esta receta no sólo crea la carpeta de la aplicación sino que añade la línea correspondiente de configuración en la variable `INSTALLED_APPS` del fichero `settings.py`.

3. Modelos



3.1. games.Game

Modelo que representa un videojuego.

Campo	Tipo
title ^(*)	str
slug ^(*)	str
description ^(∅)	str
cover ^(∅)	image
price ^(*)	float
stock ^(*)	int
released_at ^(*)	date
pegi ^(*)	enum
category ^(*)	fk → Category
platforms ^(∅)	m2m → Platform

- El **PEGI** (pegi) será un **enumerado** con valores:
 - PEGI3 = 3
 - PEGI4 = 4
 - PEGI6 = 6
 - PEGI7 = 7
 - PEGI12 = 12
 - PEGI16 = 16
 - PEGI18 = 18
- Aplica `models.PROTECT` en la clave ajena con `Category`.

3.2. games.Review

Modelo que representa una reseña de un usuario sobre un juego.

Campo	Tipo
comment ^(*)	str
rating ^(*)	int [0 – 5]
game ^(*)	fk → Game
user ^(*)	fk → User

3.3. categories.Category

Modelo que representa una categoría de videojuegos: *Aventura*, *Acción*, *Estrategia*, etc.

Campo	Tipo
name ^(*)	<i>str</i>
slug ^(*)	<i>str</i>
description ^(∅)	<i>str</i>
color ^(∅)	ColorField

3.4. platforms.Platform

Modelo que representa una plataforma de videojuegos: *PC, PS4, Xbox, etc.*

Campo	Tipo
name ^(*)	<i>str</i>
slug ^(*)	<i>str</i>
description ^(∅)	<i>str</i>
logo ^(∅)	<i>image</i>

3.5. orders.Order

Modelo que representa un pedido de un usuario.

Campo	Tipo
status ^(*)	<i>enum</i>
created_at ^(*)	<i>datetime</i>
updated_at ^(*)	<i>datetime</i>
key ^(∅)	<i>UUID</i>
user ^(*)	<i>fk</i> → User
games ^(*)	<i>m2m</i> → Game

- El estado de un pedido (**status**) será un [enumerado](#) con valores:
 - INITIATED = 0
 - CONFIRMED = 1
 - CANCELLED = 2
 - PAID = 3

3.6. users.Token

Modelo que representa un token de autenticación de un usuario.

user ^(*)	<i>o2o</i> → User
key ^(*)	<i>UUID</i>
created_at ^(*)	<i>datetime</i>

3.7. User

No hay que implementar este modelo. Se usará el modelo [User](#) que ofrece Django.

3.8. Carga de datos

Una vez que hayas creado los modelos y aplicado las migraciones, puedes cargar datos de prueba con la siguiente *receta* [just](#):

```
just load-data
```

4. URLs

Dado que estamos implementando una *API*, prácticamente todas las URLs devolverán un [JSON](#). Para poder hacer pruebas de forma sencilla, puedes utilizar la herramienta [HoppScotch](#) (gratuita y de código abierto).

4.1. `games.urls`

`/api/games/` \Rightarrow `games.views.game_list()`

Listado de los juegos disponibles en el sistema.

GET request	JSON response
	<code>game^(🕹)</code> <code>game^(🕹)</code> <code>game^(🕹)</code> <code>...</code>

`/api/games/filter?category=sports&platform=ps5` \Rightarrow `games.views.game_list()`

Listado de los juegos disponibles en el sistema filtrando por los parámetros de la petición *querystring*.

GET request	JSON response
<code>category^(🕹)</code>	<code>game^(🕹)</code>
<code>platform^(🕹)</code>	<code>game^(🕹)</code>
	<code>game^(🕹)</code>
	<code>...</code>

`/api/games/eldenring/` \Rightarrow `games.views.game_detail()`

Detalle del juego “Elden Ring”.

GET request	JSON response
	<code>title</code> <code>slug</code> <code>description</code> <code>cover</code> <code>price</code> <code>stock</code> <code>released_at</code> <code>pegi</code> <code>category</code> <code>platforms^(🕹)</code>

`/api/games/eldenring/reviews/` \Rightarrow `games.views.review_list()`

Reseñas del juego “Elden Ring”.

GET request	JSON response
	<code>review^(⌘)</code> <code>review^(⌘)</code> <code>review^(⌘)</code> <code>...</code>

`/api/games/reviews/21/` \Rightarrow `games.views.review_detail()`

Detalle de la reseña con pk=21.

GET request	JSON response
	<code>comment</code> <code>rating</code> <code>game^(⌘)</code> <code>user^(⌘)</code>

`/api/games/eldenring/reviews/add/` \Rightarrow `games.views.add_review()`

Añade una nueva reseña al juego “Elden Ring”.

POST request	JSON response
<code>token</code> <code>comment</code> <code>rating</code>	<code>pk^(review)</code>

4.2. orders.urls

`/api/orders/add/` \Rightarrow `orders.views.add_order()`

Añade un nuevo pedido.

POST request	JSON response
<code>token</code>	<code>pk^(order)</code> <code>status</code> <code>created_at</code>

`/api/orders/53/` \Rightarrow `orders.views.order_detail()`

Detalle del pedido con `pk=53`.

POST request	JSON response
token	status created_at updated_at key games ^(♡)

`/api/orders/53/confirm/` \Rightarrow `orders.views.confirm_order()`

Confirmación del pedido con `pk=53`.

POST request	JSON response
token	status created_at updated_at

`/api/orders/53/cancel/` \Rightarrow `orders.views.cancel_order()`

Cancelación del pedido con `pk=53`.

POST request	JSON response
token	status created_at updated_at

`/api/orders/53/pay/` \Rightarrow `orders.views.pay_order()`

Pago del pedido con `pk=53`.

POST request	JSON response
token	status created_at updated_at key