

Diario de Aprendizaje: Comandos y Conceptos

Aquí documentaré cada comando que ejecutamos para que entiendas **qué** estamos haciendo y **por qué**. Aunque estás en Windows, estos comandos son estándar en el desarrollo web profesional (y muy similares a Linux).

Concepto Clave: ¿Qué es Strapi? (La Metáfora del Restaurante)

Imagina un restaurante tradicional (como WordPress):

- Tienen la cocina (Backend) y las mesas (Frontend) en el mismo local.
- Si quieres cambiar la decoración, tienes que cerrar el restaurante.
- Solo sirven comida en *su* local.

Strapi es una "Dark Kitchen" (Cocina Fantasma) 🍷:

1. **Solo Cocina (Backend)**: Se dedica exclusivamente a preparar los datos (Alquileres, Usuarios, Precios). No tiene "comedores" (no tiene parte visual para el cliente final).
2. **La Ventanilla (API)**: Strapi pone la comida en una ventanilla y dice "¡Pedido listo!".
3. **El Repartidor (Astro)**: Tu web (Astro) es como un repartidor de Uber Eats. Viene a la ventanilla, coge los datos y se los lleva al cliente para montarlos en su móvil, en su ordenador o donde sea.

¿Por qué es mejor esto?

- Porque Astro (el repartidor) es **rapidísimo**.
- Porque si mañana quieres hacer una App de móvil, la App puede ir a la misma ventanilla de Strapi a por datos. ¡No tienes que crear otro backend!

Tour por la Interfaz de Strapi: Configuración Avanzada

En la pestaña "Advanced Settings" del creador de tablas:

1. Draft & Publish (Borrador y Publicado)

"El sistema de revisión".

- **Activado (Recomendado):** Cuando creas un alquiler, nace como "Draft" (Borrador). Nadie lo ve en la web hasta que le das al botón verde "Publish".
- **Desactivado:** En cuanto le das a guardar, ya es público. Es peligroso si estás editando cosas en vivo.
- **Veredicto:** ¡Déjalo activado! Es profesional.

2. Internationalization (i18n) 🌐

"La Torre de Babel".

- Sirve para tener tu web en Inglés, Español, Francés...
- Si lo activas, te dejará crear una versión del alquiler en Español y otra en Inglés.
- **Veredicto:** Para este proyecto, **desactívalo** si quieres simplificar. Si el profesor pide multi-idioma, actívalo. (Por defecto viene activado, no molesta).



Concepto Clave: El Molde vs. La Galleta

Has preguntado: "¿Qué estoy haciendo realmente?". Esta es la mejor analogía:

Estás en el **Taller del Arquitecto (Content-Type Builder)**.

- **Lo que haces ahora:** Estás fabricando el **MOLDE** de las galletas.
 - Dices: "Mis galletas serán redondas (título), tendrán pepitas de chocolate (precio) y un relleno (descripción)".
 - **Aún NO hay galletas.** Solo hay un molde de metal vacío.
 - El botón **SAVE** (arriba a la derecha) es lo que envía ese molde a la fábrica (Base de Datos) para que empiece a funcionar.
- **Lo que harás luego:** Irás al **Content Manager** (La Cocina).
 - Allí usarás el molde que acabas de crear para hacer **500 galletas reales** (Apartamento en Málaga, Ático en Cádiz, etc.).



Tour por la Interfaz de Strapi: Creando tu Colección

Estás en la pantalla de "Nacimiento de una Tabla".

Campos de Configuración

1. **Display Name:** El nombre "humano" que verás tú en el panel.
 - Ejemplo: **Rental**.
2. **API ID (Singular):** El nombre que usará el código para referirse a UN elemento.
 - Ejemplo: **rental**. (Automático).
3. **API ID (Plural):** El nombre que usará la URL para pedir la LISTA.
 - Ejemplo: **rentals**. (Automático).
 - Por esto la URL será **/api/rentals** (en plural).



Tour por la Interfaz de Strapi: Tipos de Datos

Esta es una pregunta de examen. Entender la diferencia es vital.

1. Collection Types (Colecciones) 📖

"Quiero muchos de estos".

- Úsalo cuando vas a tener una lista de cosas.
- Ejemplos: Alquileres, Productos, Usuarios, Artículos del Blog.
- **En tu caso:** **Rental** es una Collection Type (porque vas a tener el apartamento A, el B, el C...).

2. Single Types (Únicos) 1

"Solo hay uno en toda la web".

- Úsalo para páginas estáticas o configuraciones globales.
- Ejemplos: La "Home Page" (solo hay una), la página "Sobre Nosotros", el "Pie de página" (Footer).

3. Components (Piezas de LEGO) 🧩

"Esto lo voy a reutilizar dentro de otros sitios".

- No tienen vida propia. Viven *dentro* de una Collection o un Single Type.
- Ejemplos: Un botón (tiene texto + enlace), una galería de fotos, una sección de SEO.
- Lo usarás para crear estructuras repetibles.



Tour por la Interfaz de Strapi

Entender la herramienta es el 50% del trabajo. Aquí tienes tu mapa:

1. Content-Type Builder (El Arquitecto) 🏗️

Aquí es donde **DISEÑAS** tu base de datos.

- No metes datos aquí. Aquí defines *qué* datos vas a guardar.
- Es como crear las cabeceras de un Excel (Nombre, Precio, Foto).
- **Acción:** Aquí crearemos "Rental" y le diremos que tiene un campo de texto para el título y un número para el precio.

2. Content Manager (El Editor) 🖋️

Aquí es donde **ESCRIBES** el contenido.

- Una vez que el Arquitecto ha hecho los planos, vienes aquí a rellenar los datos.
- Es como rellenar las filas del Excel.

- **Acción:** Aquí entrarás a "Rental" y crearás "Apartamento en Torremolinos", "Ático en Marbella", etc.

3. Media Library (El Álbum)

Tu carpeta de imágenes y vídeos.

- Todo lo que subas (fotos de pisos, PDFs) se guarda aquí.
- Strapi es listo y te crea varios tamaños automáticamente (thumbnail, medium, large) para que tu web cargue rápido.

4. Settings (La Sala de Máquinas)

Configuración técnica y PERMISOS.

- Aquí está la llave del grifo.
- **Users & Permissions:** Lo más importante. Por defecto, Strapi no deja entrar a nadie. Aquí es donde le diremos "Deja que TODO EL MUNDO (Public) pueda VER (find) los alquileres".

5. Marketplace & Deploy (Los Extras)

- **Marketplace:** Es como la App Store de Strapi. Puedes instalar plugins (ej: uno para SEO, otro para menús). No solemos tocarlo al principio.
- **Deploy:** Son atajos para subir tu proyecto a la nube (Strapi Cloud). Como estamos aprendiendo en local, ni caso por ahora.



La Clase Maestra: URLs y la "Esencia" de la Web

Has dado en el clavo. **La URL es la verdad.** Todo en la web (PHP, Astro, Strapi, Google) depende de entender esto.

Anatomía de una URL

Analicemos la URL que te dio problemas:

`http://localhost:1337/admin/usecase?hasAdmin=true`

1. **Protocolo (`http://`):** Las reglas del juego. "Vamos a hablar en texto plano web".
2. **Dominio/Host (`localhost`):** ¿A qué ordenador llamo? `localhost` soy yo mismo.
3. **Puerto (`:1337`):** ¿A qué puerta llamo?
 - Strapi vive en la puerta 1337.
 - Astro vive en la puerta 4321.
 - Las webs normales viven en la 80 o 443 (por eso no se pone puerto en google.com).
4. **Ruta (`/admin/usecase`):** ¿Qué carpeta o archivo quiero?
 - En **PHP clásico**, esto solía ser un archivo real (`/admin/usecase.php`).
 - En **Apps Modernas (Strapi/Astro)**, esto es una **Ruta Virtual**. No existe esa carpeta. Hay un "portero" (el Router) que lee `/admin` y decide qué código ejecutar.
5. **Parámetros GET (`?hasAdmin=true`):** Aquí está la "esencia PHP" que decías.
 - Empieza con `?`.

- Es información extra que le pasamos a la página.
- `hasAdmin=true` le estaba diciendo a la página: "Oye, comportate sabiendo que YA existe un admin".

¿Por qué esto es vital para tu Proyecto?

Porque tu frontend (Astro) no va a "leer la base de datos". Va a **leer URLs**.

Cuando quieras los alquileres, Astro llamará a: `http://localhost:1337/api/rentals`

- Si le pides `http://localhost:1337/api/rentals?filters[price][$lt]=500`:
 - Le dices: "Dame los que tengan *price* menor que (\$lt) 500".

Entender la URL es entender cómo pedirle cosas a Strapi.

⚠ Solución de Problemas Comunes

"Me sale una URL rara y no avanza" (`/admin/usecase?hasAdmin=true`)

Esta es la pantalla que te tiene atrapado.

- **Qué significa:** `http://localhost:1337/admin/usecase?hasAdmin=true`
 - `/admin`: Estás en el panel de control.
 - `/usecase`: Estás en la pantalla de "Caso de uso" (la encuesta).
 - `?hasAdmin=true`: ¡Buenas noticias! Esto significa que tu usuario **SÍ** se ha creado correctamente. El sistema sabe que existe un admin.
- **El Problema:** La encuesta a veces falla al enviar los datos a Strapi (sus servidores de marketing), y por eso no te deja pasar.
- **El Truco:** Como ya eres admin, no necesitas hacer la encuesta. Simplemente borra todo lo que hay después de `/admin` en la barra de direcciones del navegador.

"No veo el panel, solo un formulario" (Creación de Admin)

La primera vez que entras a Strapi, te pide **crear el primer usuario**.

- **¿Por qué?:** Strapi es seguro. Nadie puede entrar sin llave. Tú estás "forjando la llave maestra" (el usuario Administrador).
- **Acción:** Rellena el formulario con cualquier dato (ej: `admin`, `admin@test.com`, `Password123`).
- **Resultado:** En cuanto le des al botón de abajo, ¡BOOM! Entrarás al panel de control real.

"No se puede acceder al sitio web" (Backend Caído)

Si Strapi deja de funcionar de repente, el 99% de las veces es porque el servidor se ha detenido.

- **Causa:** Cuando creas un "Content Type" nuevo, Strapi necesita reiniciar su núcleo para "entender" la nueva estructura de la base de datos. A veces, en Windows, este reinicio automático falla y el proceso muere.
- **Solución:**
 1. Vuelve a la terminal donde tenías Strapi (la del `cd backend`).

2. Pulsa **Flecha Arriba** para recuperar el último comando (o escribe `npm run develop`).
3. Dale a **Enter**.

¡Es normal! En desarrollo, los servidores se paran y arrancan constantemente. No te asustes.

Conceptos de Base de Datos y CMS

🧠 ¿Dónde está la Base de Datos?

Has preguntado si hay que crear una base de datos. La respuesta corta es: **Ya la tienes**. Al usar `--quickstart`, Strapi creó un archivo llamado `data.db` (una base de datos **SQLite**) dentro de tu carpeta `.tmp`.

- **SQLite**: Es una base de datos "de bolsillo". No necesita un servidor aparte (como MySQL o PostgreSQL). Vive en un archivo. Es perfecta para desarrollar rápido.
- **Content Types (Tipos de Contenido)**: En lugar de "Crear Tablas" (como en SQL), en Strapi creamos "Content Types".
 - Si fuera un Excel, el *Content Type* sería el nombre de la hoja (ej: "Alquileres").
 - Los *Fields* serían las columnas (ej: "Precio", "Fotos", "Dirección").

Comandos Ejecutados (Exploración y Ejecución)

4. `cd backend` (Navegación)

El comando que usaste para entrar en la carpeta.

- **Linux Tip**: ¡El espacio es vital! `cdbackend` (todo junto) no existe. `cd` es el verbo (cambiar directorio) y `backend` es el destino. Siempre van separados.
- **Truco**: Si escribes `cd bac` y pulsas **TAB**, la terminal completará el nombre `backend` sola. ¡Úsala siempre para no cometer errores de escritura!

5. `npm run develop` (Arrancar Motores)

Este comando arranca el servidor de desarrollo de Strapi.

- `npm run`: Le dice a Node "ejecuta un script que está definido en el archivo `package.json`".
- `develop`: Es el nombre del script. En Strapi, esto levanta el servidor en modo "relojería" (si cambias código, se reinicia solo) y abre el panel de administración.
- **Lo que pasa por detrás**: Node lee `package.json`, busca `"scripts": { "develop": "strapi develop" }` y ejecuta ese comando final.

6. `npm run dev` (Astro en marcha)

Muy parecido al anterior, pero para Astro.

- **dev**: Es el estándar casi universal para "arrancar en modo desarrollo".
- **Terminal Bloqueada**: Habrás notado que al ejecutar esto, la terminal se queda "pensando" y no te deja escribir más. ¡Eso es bueno! Significa que el servidor está escuchando. Si quieres pararlo, el comando universal es **Ctrl + C**.

Comandos Ejecutados (Fase de Instalación)

1. `npx create-strapi-app@latest . --quickstart --no-run`

Este comando instaló el **Backend (Strapi)** en la carpeta `backend`.

- **npx**: Ejecuta un paquete de Node.js sin tener que instalarlo globalmente. Es como "alquilar" una herramienta para usarla una vez.
- **create-strapi-app@latest**: El nombre del "paquete" que crea la aplicación Strapi. **@latest** asegura que usamos la última versión.
- **.**: Indica que queremos instalarlo en el **directorio actual** (en este caso, dentro de `backend`).
- **--quickstart**: Una "bandera" (flag) que le dice a Strapi: "Usa la configuración por defecto y SQLite". Ahorra tiempo de configuración manual.
- **--no-run**: Le dice: "Instálalo, pero no arranques el servidor todavía".

2. `npm create astro@latest . -- [opciones...]`

Este comando instaló el **Frontend (Astro)** en la carpeta `frontend`.

- **npm create**: Similar a **npx**, es una forma corta de decir "ejecuta el creador de este paquete".
- **--template basics**: Usamos una plantilla básica en lugar de una vacía.
- **--install**: Instala las dependencias (las librerías que Astro necesita) automáticamente.
- **--no-git**: No inicializamos un repositorio Git dentro de la carpeta frontend (ya tienes uno global).

3. `npx astro add tailwind --yes`

Este comando añadió **Tailwind CSS** a tu proyecto Astro.

- **astro add**: Una utilidad de Astro para instalar integraciones oficiales.
- **tailwind**: El nombre de la integración.
- **--yes**: Acepta automáticamente todas las preguntas de configuración (para hacerlo desatendido).



Rincón Linux (Conceptos de Terminal)

Aunque uses PowerShell o CMD en Windows, estos conceptos son universales en Linux/Mac:

- **cd (Change Directory)**: El comando para moverse entre carpetas.

- Ejemplo: `cd frontend` (entra a la carpeta frontend).
- Ejemplo: `cd ..` (sube un nivel, vuelve atrás).
- **Flags (--):** Las opciones que ponemos con guiones (como `--yes` o `--quickstart`) modifican el comportamiento del comando. Son como los "adjetivos" en una frase.