COSTASOL ALQUILER

Directorio de Alquiler de Larga Temporada • Costa del Sol

Estrategia y Planning • 18 Días

Stack: Astro (SSG) + Strapi (Headless CMS) + AWS Free Tier

Proyecto Final DAW • 2º Curso • Digitech Málaga • Febrero 2026

📋 RESUMEN EJECUTIVO

🏠 El proyecto

Directorio/buscador de alquiler de larga temporada en la Costa del Sol (Málaga). Zonas: Benalmádena, Torremolinos, Fuengirola, Marbella y Málaga capital. Dataset mock realista y escalable, diseñado para generar muchas páginas indexables.

⚙️ Stack técnico

Frontend: Astro en modo SSG (estático). Backend: Strapi como headless CMS con API REST. Hosting: AWS Free Tier (EC2 t2.micro). CI/CD: GitHub Actions. SSG recomendado porque genera HTML puro = Lighthouse perfecto + sitemap automático + indexación instantánea.

📅 Fases del planning (18 días)

FASE 1 (Días 1–4): Cimientos — Setup, estructura, Strapi, primer deploy. FASE 2 (Días 5–9): Core evaluable — Listados, fichas, filtrado dinámico, SEO. FASE 3 (Días 10–13): Pulido — Métricas, Search Console, Lighthouse, UX. FASE 4 (Días 14–16): Preparación examen — Practicar JS sin IA. FASE 5 (Días 17–18): Cierre — Deploy final, defensa oral.

¿Por qué SSG y no SSR?

- **Simplicidad:** Astro SSG genera HTML puro en build-time. Webempresa, AWS, Vercel... cualquier servidor puede servirlo.
- **Performance:** HTML estático = Lighthouse >95 casi automático. No hay servidor renderizando.
- **SEO:** El sitemap se genera en build con @astrojs/sitemap. Cada página estática = 1 URL indexable.
- **Hosting:** No necesitas Node.js corriendo 24/7 en el servidor de frontend. Solo Strapi necesita Node.
- **Fiabilidad:** Menos cosas que pueden fallar. En 18 días, cada hora cuenta.

🧱 FASE 1: CIMIENTOS (DÍAS 1–4)

Objetivo de fase: Tener el proyecto Astro + Strapi funcionando, con datos mock cargados, y el primer deploy online accesible por URL. Sin esto, nada de lo demás importa.

DÍA 1 • FASE 1 • CIMIENTOS

🎯 **Objetivo:** Crear proyecto Astro, estructura de carpetas y repositorio GitHub

Tareas:

- Instalar Astro: npm create astro@latest costasol-alquiler
- Estructura: /src/pages/, /src/components/, /src/layouts/, /src/data/, /src/styles/
- Instalar Tailwind: npx astro add tailwind
- Crear Layout base (Header + Footer + slot para contenido)
- Crear página index.astro con contenido placeholder
- Inicializar Git + primer commit + push a GitHub

✅ **Definition of Done:** npm run dev funciona. Repo en GitHub con primer commit. Layout base visible en localhost:4321.

⚠️ **Riesgo:** Versión de Node incompatible → Usar nvm para instalar Node 20 LTS.

DÍA 2 • FASE 1 • CIMIENTOS

🎯 **Objetivo:** Instalar Strapi, definir Content Types y cargar datos mock iniciales

Tareas:

- Instalar Strapi: npx create-strapi@latest costasol-cms (SQLite para desarrollo)
- Crear Content Type 'Propiedad': titulo, slug, descripcion, precio, zona, tipo (piso/casa/estudio), habitaciones, baños, superficie, imagen_url, disponible, destacado
- Crear Content Type 'Zona': nombre, slug, descripcion, imagen_url, municipio
- Cargar 20–30 propiedades mock realistas distribuidas por las 5 zonas
- Activar permisos públicos en Strapi: GET /api/propiedades y GET /api/zonas
- Verificar que la API responde en http://localhost:1337/api/propiedades

✅ **Definition of Done:** Strapi corriendo en localhost:1337. API pública devuelve JSON con 20+ propiedades y 5 zonas.

⚠️ **Riesgo:** Strapi falla al instalar con Node 21+ → Usar Node 20 LTS. Verificar con node -v antes de instalar.

DÍA 3 • FASE 1 • CIMIENTOS

🎯 **Objetivo:** Conectar Astro con Strapi: fetch de datos en build-time y generar páginas estáticas

Tareas:

- Crear /src/lib/strapi.ts con función fetchAPI(endpoint) que haga fetch a Strapi
- En index.astro: fetch propiedades destacadas y renderizar grid de tarjetas
- Crear /src/pages/propiedades/[slug].astro con getStaticPaths() para generar fichas
- Crear /src/pages/zonas/[slug].astro con getStaticPaths() para páginas de zona

• Ejecutar npm run build y verificar que genera HTML para TODAS las rutas
• Comprobar que dist/ contiene /propiedades/piso-2hab-benalmadena/index.html etc.

✅ **Definition of Done:** npm run build genera dist/ con páginas estáticas para cada propiedad y zona. Sin errores de build.

⚠️ **Riesgo:** Fetch falla porque Strapi no está corriendo → Añadir script que arranca Strapi antes del build. O usar datos mock JSON como fallback.

DÍA 4 • FASE 1 • CIMIENTOS

🎯 **Objetivo:** Primer deploy real: subir dist/ al servidor y tener URL accesible online

Tareas:
• OPCIÓN AWS: Lanzar EC2 t2.micro, instalar Nginx, subir dist/ por SCP
• OPCIÓN WEBEMPRESA: Subir dist/ por FTP al public_html
• OPCIÓN VERCEL: Conectar repo GitHub y deploy automático
• Configurar subdominio apuntando al servidor
• Verificar que la web carga correctamente en el navegador desde la URL pública
• Configurar GitHub Action básica para deploy automático en cada push

✅ **Definition of Done:** La web es accesible por URL pública (subdominio). Se ve el listado de propiedades y puedes navegar a fichas.

⚠️ **Riesgo:** Problemas de DNS (tarda 24–48h en propagar) → Usar IP directa mientras propaga. O usar subdominio de Vercel si el profesor lo acepta.

⭐ FASE 2: CORE EVALUABLE (DÍAS 5–9)

Objetivo de fase: Tener listados completos, fichas de propiedad, filtrado dinámico en JavaScript, SEO técnico básico. Esto es LO QUE EL PROFESOR EVALÚA. Sin esta fase, no apruebas.

DÍA 5 • FASE 2 • CORE EVALUABLE

🎯 **Objetivo:** Diseñar y maquetar listado de propiedades con grid responsiva

Tareas:

- Componente PropertyCard.astro: imagen, título, zona, precio, habitaciones, superficie
- Grid responsiva con Tailwind: 1 col móvil, 2 cols tablet, 3 cols desktop
- Página /propiedades/ con todas las propiedades en grid
- Páginas de zona /zonas/benalmadena/ etc. con propiedades filtradas por zona
- Breadcrumbs: Inicio > Zonas > Benalmádena > Piso 2 habitaciones...
- Deploy y verificar online

✅ **Definition of Done:** Páginas de listado y zona se ven bien en móvil y desktop. Grid responsiva funcionando.

⚠️ **Riesgo:** Tarjetas se descuadran en móvil → Usar grid con minmax(280px, 1fr) en lugar de columnas fijas.

DÍA 6 • FASE 2 • CORE EVALUABLE

🎯 **Objetivo:** Maquetar fichas individuales de propiedad con todos los datos

Tareas:

- Página /propiedades/[slug].astro completa: imagen hero, datos, descripción, mapa zona
- Sección de características: habitaciones, baños, superficie, tipo, precio
- Botón CTA (mock): "Contactar" o "Solicitar visita"
- Componente RelatedProperties.astro: 3 propiedades similares (misma zona o tipo)
- Links de navegación: volver a zona, volver a listado
- Deploy y verificar fichas online

✅ **Definition of Done:** Cada propiedad tiene su página individual con URL amigable (/propiedades/piso-2hab-benalmadena). Datos completos visibles.

⚠️ **Riesgo:** Slugs duplicados si dos pisos tienen mismo nombre → Añadir ID al slug (piso-2hab-benalmadena-001).

DÍA 7 • FASE 2 • CORE EVALUABLE

🎯 **Objetivo:** Implementar filtrado dinámico con JavaScript vanilla en el frontend

Tareas:

- Crear /src/scripts/filtros.js con lógica de filtrado client-side
- Filtros: por zona (select), por tipo (piso/casa/estudio), por rango de precio (slider o inputs), por habitaciones (botones)
- Los filtros operan sobre los datos del DOM (ya renderizados en HTML por Astro SSG)

- Cada cambio de filtro actualiza la grid instantáneamente sin recargar página
- Mostrar contador: "X propiedades encontradas"
- Después del filtrado, añadir <script> tag en la página de listado con los datos JSON inline

✅ **Definition of Done:** Los filtros funcionan en /propiedades/: seleccionar zona + tipo + precio muestra solo propiedades que cumplen TODOS los criterios simultáneamente.

⚠️ **Riesgo:** Los datos no están en el DOM → Inyectar JSON en un <script type="application/json"> dentro de la página Astro y leerlo con JS.

DÍA 8 • FASE 2 • CORE EVALUABLE

🎯 **Objetivo:** SEO técnico básico: sitemap, robots.txt, slugs y metadatos

Tareas:

- Instalar @astrojs/sitemap: npx astro add sitemap. Configurar en astro.config.mjs
- Crear public/robots.txt: Allow para páginas, Disallow para /admin/ y /api/
- Verificar que TODOS los slugs son amigables: /propiedades/piso-2hab-benalmadena, /zonas/fuengirola
- Añadir <title>, <meta description>, <meta og:*> únicos por página (componente SEO.astro)
- Añadir canonical URL a cada página. Si hay filtros con parámetros URL, noindex o canonical a la base
- Build + deploy + verificar sitemap.xml y robots.txt en la URL pública

✅ **Definition of Done:** tudominio/sitemap.xml devuelve XML válido con TODAS las URLs. robots.txt accesible. Cada página tiene title + description únicos.

⚠️ **Riesgo:** Sitemap no incluye páginas dinámicas → Verificar que getStaticPaths() genera todas las rutas antes del build.

DÍA 9 • FASE 2 • CORE EVALUABLE

🎯 **Objetivo:** Escalar dataset: 50–100 propiedades + dar de alta en Search Console

Tareas:

- Ampliar datos mock en Strapi: mínimo 50 propiedades realistas distribuidas por zonas
- Variedad: distintos precios (600–2500€), tipos, superficies, zonas, estados
- Rebuild + deploy (cada propiedad nueva = nueva página estática = nueva URL indexable)
- Dar de alta el sitio en Google Search Console. Enviar sitemap.xml
- Verificar propiedad en Search Console (método DNS, meta tag o archivo HTML)
- Primer test Lighthouse: anotar puntuaciones base de Performance, SEO, Accessibility, Best Practices

✅ **Definition of Done:** 50+ propiedades online. Search Console verificado y sitemap enviado. Puntuaciones Lighthouse anotadas.

⚠️ **Riesgo:** Search Console tarda en verificar DNS → Usar método meta tag (instantáneo) como alternativa.

✨ FASE 3: PULIDO Y MÉTRICAS (DÍAS 10–13)

Objetivo de fase: Optimizar UX, medir métricas reales, añadir valor añadido. Aquí es donde subes de notable a sobresaliente.

DÍA 10 • FASE 3 • PULIDO

🎯 **Objetivo:** Optimizar UX: responsive perfecto, loading states, 404 personalizado

Tareas:

- Revisar TODA la web en móvil (iPhone SE), tablet y desktop
- Crear página 404.astro personalizada con buscador y links útiles
- Añadir estados vacíos al filtrado: "No hay propiedades con estos criterios" con sugerencia
- Optimizar imágenes: formatos WebP, lazy loading, aspect-ratio definido
- Añadir favicon y Open Graph images para compartir en redes
- Deploy + verificar en móvil real (no solo DevTools)

✅ **Definition of Done:** La web se ve perfecta en móvil real. 404 personalizado. Imágenes optimizadas. Sin layout shifts.

⚠️ **Riesgo:** Imágenes pesadas bajan Lighthouse → Usar Astro <Image> component o placeholder con aspect-ratio fijo.

DÍA 11 • FASE 3 • PULIDO

🎯 **Objetivo:** Métricas: segundo Lighthouse, Search Console, documentar evolución

Tareas:

- Ejecutar Lighthouse en 3 páginas clave: home, listado, ficha. Anotar puntuaciones
- Revisar Search Console: ¿se han indexado páginas? ¿Errores de rastreo?
- Crear documento de métricas (Excel o tabla en el README) con datos semana 1 vs semana 2
- Corregir cualquier error que reporte Lighthouse o Search Console
- Añadir structured data (JSON-LD) básico: tipo RealEstateListing en fichas de propiedad
- Deploy con correcciones

✅ **Definition of Done:** Lighthouse >90 en las 4 categorías para las 3 páginas clave. Métricas documentadas en tabla comparativa.

⚠️ **Riesgo:** Lighthouse baja por third-party scripts → No añadir Google Analytics aún. Usar solo Search Console.

DÍA 12 • FASE 3 • PULIDO

🎯 **Objetivo:** Valor añadido: búsqueda por texto, ordenación, mapa de zonas

Tareas:

- Añadir buscador por texto libre: filtra por título, zona y descripción
- Añadir ordenación: precio asc/desc, superficie, más recientes
- Crear página /zonas/ con mapa visual de la Costa del Sol (puede ser SVG o imagen con links)
- Añadir estadísticas por zona: nº propiedades, precio medio, rango de precios

- Mejorar formulario de contacto mock con normalización (trim, toLowerCase del email)
- Deploy y verificar todo online

✅ **Definition of Done:** Búsqueda y ordenación funcionan combinados con los filtros existentes. Página de zonas con mapa visual.

⚠️ **Riesgo:** Buscador lento con muchas propiedades → Debounce de 300ms en el input para no filtrar en cada tecla.

DÍA 13 • FASE 3 • PULIDO

🎯 **Objetivo:** Pulido final: accesibilidad, performance, detalles visuales

Tareas:

- Auditoría de accesibilidad: alt en imágenes, contraste, focus visible, aria-labels
- Añadir animaciones sutiles: transiciones en filtros, hover en tarjetas
- Revisar y mejorar la paleta cromática: coherencia visual completa
- Asegurar que TODAS las páginas tienen breadcrumbs + navegación clara
- Añadir schema.org markup para BreadcrumbList en todas las páginas
- Deploy final de Fase 3. Anotar Lighthouse definitivo.

✅ **Definition of Done:** Lighthouse >95 en Performance y SEO. Accesibilidad >90. Web visualmente pulida y profesional.

⚠️ **Riesgo:** Animaciones CSS causan jank → Usar solo transform y opacity (GPU-accelerated). Nada de animate width/height.

⌨️ FASE 4: PREPARACIÓN EXAMEN JS (DÍAS 14–16)

💣 RECORDATORIO: El examen final PRÁCTICO PERMANECE

El proyecto elimina el examen del viernes 20, pero el examen final de JS + Interfaces SIGUE. Tienes 2 horas para demostrar desde cero: grid, render, filtrado, formulario. SIN IA. Si no lo practicas, el 30% de la nota te hunde.

DÍA 14 • FASE 4 • EXAMEN

🎯 **Objetivo:** Práctica examen 1: Render dinámico + grid desde cero (cronómetro 2h)

Tareas:

- Crear carpeta /practica-examen/ FUERA del proyecto Astro (HTML + CSS + JS puros)
- Cronómetro: 2 horas. JSON de propiedades dado. SIN IA, SIN copiar del proyecto.
- Construir grid responsiva con CSS Grid/Flexbox desde cero
- Renderizar propiedades del JSON en el DOM con .map() + innerHTML/createElement
- Implementar desestructuración al extraer datos del JSON
- Al acabar: revisar qué falló, qué costó más tiempo, qué olvidaste

✅ **Definition of Done:** En 2 horas has montado una grid con datos renderizados dinámicamente desde JSON. Sin ayuda externa.

⚠️ **Riesgo:** Te bloqueas en .map() o createElement → Practica estos métodos aislados antes del simulacro. Deben ser automáticos.

DÍA 15 • FASE 4 • EXAMEN

🎯 **Objetivo:** Práctica examen 2: Filtrado + formulario + estado (cronómetro 2h)

Tareas:

- Mismo setup: HTML + CSS + JS puros, cronómetro 2h, SIN IA
- Añadir filtros: por tipo (select), por precio (rango), por zona (botones)
- Cada filtro actualiza la grid en tiempo real con .filter() encadenado
- Añadir formulario: nombre + email + mensaje. Capturar con addEventListener('submit')
- Normalizar datos: trim(), toLowerCase() en email, validación básica
- Añadir nuevo dato al estado y re-renderizar el DOM

✅ **Definition of Done:** Filtrado funcional combinado + formulario que captura datos, los normaliza y actualiza el DOM. En <2h.

⚠️ **Riesgo:** Los filtros no se combinan bien → Estrategia: aplicar TODOS los filtros en cadena sobre el array original, nunca sobre el array ya filtrado.

DÍA 16 • FASE 4 • EXAMEN

🎯 **Objetivo:** Simulacro completo + repaso de conceptos clave

Tareas:

- Simulacro COMPLETO: grid + render + filtrado + formulario en 2 horas

- Cronometrar cada fase: ¿cuánto tardas en la grid? ¿en el render? ¿en filtros?
- Repasar conceptos que te costarán en la defensa: .map(), .filter(), .find(), .reduce()
- Repasar desestructuración, spread operator, template literals
- Escribir en papel (sí, papel) las funciones clave de filtrado sin mirar código
- Preparar el HTML base + Tailwind + JSON que puedes llevar preparado al examen

✅ **Definition of Done:** Has completado el simulacro completo en <2h sin errores críticos. Puedes explicar cada línea.

⚠️ **Riesgo:** Te pasas de 2h → Identifica el cuello de botella (suele ser el filtrado combinado) y practica ESO en bucle.

🚀 FASE 5: CIERRE Y DEFENSA (DÍAS 17–18)

DÍA 17 • FASE 5 • CIERRE

🎯 **Objetivo:** Deploy final limpio + documentación + README profesional

Tareas:

- Rebuild final con TODOS los datos y correcciones
- Deploy limpio: verificar que TODO funciona en la URL pública
- Escribir README.md profesional: descripción, stack, screenshots, cómo ejecutar
- Documentar métricas finales: Lighthouse, nº páginas indexadas, Search Console
- Revisar que el repositorio GitHub está limpio: .gitignore, commits descriptivos, sin secretos
- Último commit: "v1.0 — Versión final para entrega"

✅ **Definition of Done:** Web online y funcionando. README completo. Repo limpio. Métricas documentadas. Listo para entregar.

⚠️ **Riesgo:** Bug de última hora → NO añadas features nuevas hoy. Solo corrige bugs y despliega.

DÍA 18 • FASE 5 • CIERRE

🎯 **Objetivo:** Preparar y ensayar defensa oral (5–7 minutos)

Tareas:

- Escribir guión de defensa oral (ver sección al final del documento)
- Ensayar 3 veces cronómetro en mano: 5–7 minutos exactos
- Preparar respuestas a preguntas frecuentes: ¿por qué Astro? ¿por qué SSG? ¿por qué este nicho?
- Tener abiertos: web desplegada, repo GitHub, Search Console, Lighthouse
- Practicar explicar una función de filtrado línea por línea
- Descansar bien la noche anterior. Confía en el trabajo hecho.

✅ **Definition of Done:** Puedes presentar el proyecto en 5–7 minutos, explicar cualquier línea de código, y defender tus decisiones.

⚠️ **Riesgo:** Nervios → La mejor cura es el ensayo. Si lo has ensayado 3 veces, los nervios bajan un 70%.

✅ CHECKLIST: PRODUCCIÓN LISTA

Antes de entregar, verifica que TODOS estos puntos se cumplen:

- ✅ Web accesible por URL pública (subdominio o dominio propio)
- ✅ TODAS las páginas cargan sin errores 404 ni consola roja
- ✅ Grid de propiedades responsiva: móvil (1 col), tablet (2 col), desktop (3 col)
- ✅ Fichas individuales con URL amigable: /propiedades/nombre-slug
- ✅ Filtrado dinámico funcional: zona + tipo + precio combinados
- ✅ Formulario de contacto captura datos y los normaliza
- ✅ Sitemap.xml accesible en /sitemap-index.xml o /sitemap-0.xml
- ✅ Robots.txt accesible en /robots.txt con reglas correctas
- ✅ HTTPS activo (certificado SSL válido)
- ✅ Repositorio GitHub público con historial de commits real
- ✅ README.md profesional con descripción, stack, instrucciones
- ✅ Sin secretos expuestos en el código (API keys, passwords)
- ✅ Deploy automático configurado (GitHub Actions o Vercel)
- ✅ Métricas documentadas: Lighthouse + Search Console
- ✅ 50+ propiedades en el dataset (50+ páginas indexables)

🔍 CHECKLIST: SEO TÉCNICO COMPLETO

✅ Sitemap XML generado con @astrojs/sitemap, incluye TODAS las URLs estáticas
✅ Robots.txt: Allow: / para Googlebot. Disallow: /admin/, /api/ si aplica
✅ Slugs amigables en TODAS las rutas: /propiedades/piso-2hab-benalmadena, /zonas/fuengirola
✅ Cada página tiene <title> único y descriptivo (máx 60 chars)
✅ Cada página tiene <meta name="description"> único (máx 155 chars)
✅ Canonical URL (<link rel="canonical">) en cada página apuntando a sí misma
✅ Si los filtros generan URLs con parámetros (?zona=X&tipo=Y): noindex o canonical a la base
✅ Open Graph tags: og:title, og:description, og:image, og:url en todas las páginas
✅ Estructura semántica HTML: <header>, <main>, <nav>, <article>, <footer>
✅ Un solo <h1> por página. Jerarquía correcta: H1 > H2 > H3
✅ Imágenes con alt descriptivo ("Piso de 2 habitaciones en Benalmádena con vistas al mar")
✅ JSON-LD structured data: RealEstateListing en fichas, BreadcrumbList en todas
✅ Sitio verificado en Google Search Console. Sitemap enviado.
✅ Sin páginas huérfanas (todas accesibles desde la navegación)
✅ Sin enlaces rotos internos (verificar con Lighthouse o herramienta de crawling)

📊 MÉTRICAS SEMANALES A REPORTAR

Métrica	Semana 1	Semana 2	Final	Objetivo
Lighthouse Performance	(anotar)	(anotar)	(anotar)	>95
Lighthouse SEO	(anotar)	(anotar)	(anotar)	>95
Lighthouse Accessibility	(anotar)	(anotar)	(anotar)	>90
Lighthouse Best Practices	(anotar)	(anotar)	(anotar)	>90
Páginas indexadas (Search Console)	(anotar)	(anotar)	(anotar)	50+
Errores de rastreo	(anotar)	(anotar)	(anotar)	0
Sitemap URLs	(anotar)	(anotar)	(anotar)	60+
Propiedades en dataset	(anotar)	(anotar)	(anotar)	50+
Zonas cubiertas	(anotar)	(anotar)	(anotar)	5+
Tiempo carga (home)	(anotar)	(anotar)	(anotar)	<1s
Commits en GitHub	(anotar)	(anotar)	(anotar)	30+

> 💡 **Consejo**
>
> Anota las métricas CADA semana en una tabla. Poder mostrar la evolución (semana 1: Lighthouse 78 → semana 3: Lighthouse 98) impresiona muchísimo al profesor y demuestra proceso de mejora continua.

🎙️ GUIÓN DE DEFENSA ORAL (5–7 MINUTOS)

Minuto	Qué enseñar	Qué decir / justificar
0:00–0:30	Presentación personal	"Soy Antonio, he creado CostaSol Alquiler, un directorio de alquiler de larga temporada en la Costa del Sol."
0:30–1:30	Demo en vivo: navegar la web	Enseñar home > listado > filtros > ficha. "Todo lo que ves es HTML estático generado por Astro desde los datos de Strapi."
1:30–2:30	Filtrado dinámico en acción	Activar filtros en directo. "El filtrado es JavaScript vanilla, opera sobre los datos inyectados en el HTML durante el build. No hay llamadas a servidor."
2:30–3:30	Arquitectura técnica	Mostrar diagrama: Strapi (datos) → Astro SSG (build) → HTML estático → servidor. "Elegí SSG porque genera HTML puro, perfecto para SEO e indexación."
3:30–4:30	SEO + Métricas	Abrir Search Console: páginas indexadas. Abrir Lighthouse: >95. "Estas métricas demuestran que el proyecto funciona en producción real."
4:30–5:30	Código: explicar función clave	Abrir filtros.js. Explicar línea por línea: .filter(), desestructuración, actualización del DOM.
5:30–6:00	Valor añadido + cierre	"Usé Astro (framework moderno que no se da en clase), Strapi (CMS headless), deploy en AWS con CI/CD. Este proyecto me sirve de portafolio real para mis prácticas."
6:00–7:00	Preguntas del profesor	Tener abiertos: código, consola, Search Console. Responder con calma y seguridad.

Preguntas que el profesor puede hacer:

- "¿Por qué elegiste Astro y no React/Next.js?" → Porque genera HTML estático puro, ideal para SEO. React necesita hidratación y complica la indexación.
- "¿Por qué SSG y no SSR?" → Porque los datos no cambian en tiempo real (son mock). SSG = build una vez, servir siempre. Más rápido y simple.
- "¿Qué hace esta línea?" (señalando .filter()) → Explicar: filtra el array original, devuelve nuevo array con solo los elementos que cumplen la condición.
- "¿Qué pasa si añades 1000 propiedades?" → El build tarda más, pero el rendimiento en cliente no cambia. Astro genera una página HTML por propiedad.
- "¿Cómo funciona el sitemap?" → @astrojs/sitemap recorre todas las rutas generadas por getStaticPaths() y crea sitemap.xml automáticamente.
- "¿El filtrado llama al servidor?" → No. Los datos están incrustados en el HTML como JSON. JavaScript filtra en el navegador del usuario.

⚡ SIGUIENTE ACCIÓN: TAREAS PARA HOY

1️⃣ DECIDIR hosting ahora mismo

AWS Free Tier (máximo portafolio), Vercel (máxima simplicidad), o Webempresa + Render (aprovecha lo que tenéis). Pregunta al profesor qué tipo de subdominio acepta. Sin esta decisión, no puedes avanzar con el Día 4.

2️⃣ INSTALAR Astro y Strapi en local

Ejecuta: npm create astro@latest costasol-alquiler && npx create-strapi@latest costasol-cms. Verifica que ambos corren sin errores. Esto es el Día 1 + inicio del Día 2.

3️⃣ CREAR repo GitHub y primer commit

Inicializa el repo, haz el primer commit con la estructura base de Astro. Esto arranca el historial de commits que el profesor quiere ver. El reloj de 18 días empieza a correr.

¡A por el 10! 🏠💪