COLEGIO OFICIAL DE APAREJADORES Y ARQUITECTOS TÉCNICOS DE MADRID

ANGULAR

«A través del nuevo Portal de trámites digitales los colegiados obtienen información y orientación muy valiosa para su desempeño profesional de forma mucho más fácil, rápida e intuitiva.»

COOATM

Desarrollo del Portal de Trámites Digitales con Angular 8

El Colegio Oficial de Aparejadores y Arquitectos Técnicos de Madrid, COAATM, facilita la actividad profesional de sus colegiados. Cuenta con profesionales muy especializados que asesoran, acompañan y apoyan a sus colegiados en cada etapa de su vida profesional. Dispone de departamentos de Asesoría Técnica, Jurídica, Tecnológica y un Gabinete de Orientación Profesional. A través de estos servicios del COAATM, los colegiados obtienen información y orientación muy valiosa para su desempeño laboral.

El Colegio disponía de un Portal web de tramitación digital cuya arquitectura, basada en ASP.NET con un framework .NET 2.0 con Web Forms, dificultaba su evolución.

El volumen necesario de las modificaciones derivadas fundamentalmente de mejoras sustanciales en la usabilidad y la adaptabilidad a los diferentes dispositivos de visualización, unido a la antigüedad de la arquitectura tecnológica existente, obligó al Colegio a un cambio tecnológico global.

Definición del producto y entorno tecnológico

Grupo ICA diseñó un nuevo Portal web de Trámites Digitales que abarcaba no solo la adecuación del diseño a la imagen de marca existente en el portal corporativo y adaptable a dispositivos móviles, sino una reforma en profundidad de la usabilidad de las operativas colegiales respecto a los trámites digitales, fruto de la experiencia recabada en los últimos años.

La nueva arquitectura utilizada se basa en una aplicación de una sola página (Single Page Application en inglés o ***SPA***) para la capa de presentación y un ***API Rest*** como componente en el servidor que será el encargado de procesar y devolver los datos requeridos por la capa de presentación.

Tanto SPA como Api Rest son conceptos genéricos, que no van asociados a ninguna tecnología en particular. En este proyecto, la solución propuesta fue utilizar ***Angular 8*** para la capa de la organización y ***ASP.Net MVC 5 Web Api*** para la capa de servidor.

Ventajas de la solución propuesta

1. Separación organización/servidor

Al ser sistemas independientes, (se comunican con un lenguaje de intercambio como JSON) posibilitan el desarrollo de proyectos autónomos. Es transparente para el Colegio como se ha construido la API y también lo es para el servidor la utilidad que vayan a proporcionar a los datos suministrados.

Si se necesita evolucionar o refactorizar uno de los dos, back o front, se puede hacer de manera separada, siempre que se mantenga la interfaz del API.

Es posible hacer diferentes frontales con un único backend; no tiene porqué ser solo web, puede ser una app para Android otro para un App iOS, etc.

2. Independencia de tecnologías / lenguajes

Se puede desarrollar en cualquier tipo de tecnología o lenguaje que encaje con las necesidades del proyecto y del Colegio.

3. Fiabilidad, escalabilidad, flexibilidad

Con esta arquitectura, solo hay que preocuparse de que el nexo organización/servidor sea correcto; permite realizar cambios en el servidor (lenguaje, base de datos, plataforma…) siempre que se respete el 'contrato' entre organización y servidor.

Es mucho más flexible, puesto que permite alojar las páginas del front y las API Rest en servidores independientes. Esta arquitectura es escalable, ya que al ser REST un 'protocolo' sin estado, a la hora de escalar horizontalmente no hay que preocuparse de que todas las peticiones de una 'sesión' se hagan al mismo servidor, ni de implementar un mecanismo de sesión compartido entre los diferentes servidores.

4. Experiencia de usuario

No es una ventaja específica del desarrollo basado en REST, sino del uso de Ajax (peticiones asíncronas al servidor). La respuesta a solicitudes al servidor son datos planos (menor tiempo de transferencia), modificando exclusivamente las partes de la página afectada, obteniendo con ello una experiencia de usuario similar a la proporcionada por aplicaciones de escritorio.

5. Menos recursos

Al no mantener el estado de sesión en el servidor, no se requiere consumo de memoria, lo cual implica que se pueden atender muchas más peticiones. El servidor recibe una petición, la procesa y devuelve un resultado sin importar el histórico de peticiones previas. Adicionalmente, al devolver datos planos y no tener que generar el html para la respuesta, se genera menos procesamiento en servidor y por tanto son necesarios menos recursos.

Ver proyecto