**1. Arquitectura general del proyecto**

**a. División en dos módulos principales**

1. **Sitio web de la agencia (front-end público):**
   * **Objetivo:** Mostrar los servicios, portfolio (incluyendo la tienda de pádel como caso de éxito), blog, testimonios, etc.
   * **Características:** Diseño responsive, experiencia de usuario atractiva, sección de contacto, integración con redes sociales, etc.
2. **Panel de administración (back-office):**
   * **Objetivo:** Permitir al administrador gestionar los clientes, ver detalles de los servicios contratados, gestionar enlaces a proyectos (como la tienda de pádel), actualizar contenido, etc.
   * **Características:** Acceso seguro mediante autenticación y roles, interfaz dinámica para consultar y modificar datos, posibilidad de gestionar la información que se muestra en el sitio público.

**b. Integración de la tienda de pádel**

* **Caso de estudio o proyecto real:** La tienda de pádel se presenta como uno de los proyectos desarrollados para un cliente. Puedes incluir en el panel de administración una sección en la que se detallen las tecnologías utilizadas, la funcionalidad y, por supuesto, la URL para acceder a la tienda.
* **Integración:**
  + **Opción 1:** Mantener la tienda como una aplicación separada (por ejemplo, en un subdominio) y simplemente vincularla desde el sitio de la agencia.
  + **Opción 2:** Integrarla de forma modular dentro del mismo proyecto, de modo que compartan autenticación o ciertos recursos (siempre que tenga sentido para la estructura del TFG).

**2. Tecnologías recomendadas**

Dado que ya tienes experiencia con **ASP.NET C# usando MVC**, podrías aprovechar esa base y, al mismo tiempo, introducir algunas tecnologías modernas para enriquecer el proyecto. Algunas combinaciones podrían ser:

**Backend**

* **ASP.NET Core MVC / Web API:**
  + **Ventajas:**
    - Te permite desarrollar tanto la parte del sitio público como las APIs necesarias para el panel de administración.
    - Es multiplataforma y tiene gran rendimiento.
    - Puedes reutilizar parte del código de tu tienda de pádel o refactorizarla para que se integre mejor.
  + **Consideración:** Puedes implementar servicios RESTful para que el front-end (si optas por frameworks modernos) consuma datos dinámicamente.
* **Entity Framework Core:**
  + Para la capa de acceso a datos, especialmente si usas bases de datos relacionales (por ejemplo, SQL Server o PostgreSQL).
* **ASP.NET Identity:**
  + Para gestionar la autenticación y autorización en el panel de administración.

**Frontend**

Tienes varias opciones según qué tan “moderno” o SPA (Single Page Application) quieras que sea el front-end:

* **Opción tradicional con Razor + Bootstrap + JavaScript:**
  + **Ventajas:**
    - Si ya estás familiarizado con ASP.NET MVC, es una buena opción para el sitio público.
    - Bootstrap te ayudará a tener un diseño responsive sin mucho esfuerzo.
* **Opción SPA (React, Angular o Vue):**
  + **Ventajas:**
    - Permite una experiencia de usuario más dinámica, sobre todo en el panel de administración.
    - Te da la oportunidad de mostrar conocimientos en frameworks modernos.
  + **Implementación:**
    - Puedes desarrollar el panel de administración como una SPA que consuma un API REST desarrollada en ASP.NET Core.
    - Otra opción es desarrollar ambos (sitio público y panel admin) como SPA, pero esto implicaría un mayor esfuerzo en la parte de SEO para el sitio público (aunque se puede solventar con técnicas como Server-Side Rendering o prerendering).

**Otras herramientas y prácticas modernas**

* **Docker:**
  + Para containerizar la aplicación, lo que puede ser un plus en términos de despliegue y escalabilidad.
* **CI/CD (por ejemplo, con GitHub Actions o Azure DevOps):**
  + Muestra buenas prácticas de integración y despliegue continuo.
* **Servicios en la nube:**
  + Puedes desplegar la solución en **Azure**, **AWS** o **Google Cloud** para demostrar conocimientos de DevOps y escalabilidad.
* **CMS o Headless CMS (opcional):**
  + Si deseas que el contenido del sitio de la agencia sea fácilmente editable, podrías considerar integrar un CMS. Algunas opciones en el ecosistema .NET son **Orchard Core** o **Umbraco**. Sin embargo, esto podría ampliar el alcance del proyecto, por lo que dependerá del tiempo y los objetivos del TFG.

**3. Ideas para la integración y presentación del TFG**

* **Documenta el flujo de trabajo:**
  + Explica claramente cómo el proyecto se divide en módulos (sitio público, panel admin, proyecto de la tienda de pádel).
  + Describe el proceso de integración de un proyecto “cliente” (la tienda) en el portfolio de la agencia.
* **Caso práctico:**
  + Simula que la agencia tiene varios clientes, de los cuales la tienda de pádel es uno. En el panel de administración se puede ver un listado de clientes, con detalles como:
    - Nombre del cliente.
    - Servicios contratados (desarrollo web, marketing digital, etc.).
    - Enlace a la página del proyecto (por ejemplo, la tienda de pádel).
  + Esto te permite mostrar habilidades en la gestión de datos y en la presentación de información.
* **Interacción entre módulos:**
  + Puedes implementar, por ejemplo, un dashboard donde el administrador vea estadísticas o un resumen de los servicios activos, lo que añade valor a la parte administrativa.
* **Desafíos y soluciones:**
  + Expón en tu TFG los retos que encontraste al integrar dos proyectos (el anterior de la tienda y la nueva aplicación de la agencia) y cómo los resolviste (por ejemplo, refactorización de código, integración de sistemas, gestión de autenticación unificada, etc.).

**4. Consideraciones finales**

* **Cohesión en la experiencia de usuario:**
  + Asegúrate de que tanto el sitio público de la agencia como el panel de administración tengan un diseño coherente y profesional.
  + Puedes utilizar un sistema de diseño (design system) que sirva para ambos entornos, facilitando el mantenimiento.
* **Escalabilidad:**
  + Aunque se trate de un proyecto de TFG, demuestra que piensas en una solución escalable y mantenible, por ejemplo, mediante separación de concerns, buenas prácticas de programación y uso de patrones de diseño.
* **Documentación y testing:**
  + Incluye documentación tanto técnica (arquitectura, diagramas, flujos) como de usuario (manuales de uso del panel, etc.).
  + Incorpora pruebas unitarias o de integración en las partes críticas de la aplicación para resaltar tu compromiso con la calidad del software.