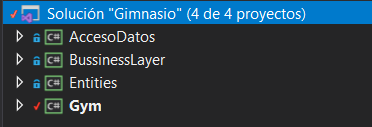
|  |  |
| --- | --- |
| Manual técnico  Práctica Profesionalizante II | Descripción  Acá se detalla todo el código escrito para la creación del programa. Hecho en 4 capas, con acceso a una base de datos local.  Alumno:  Joel Pallero |

**Manual técnico**

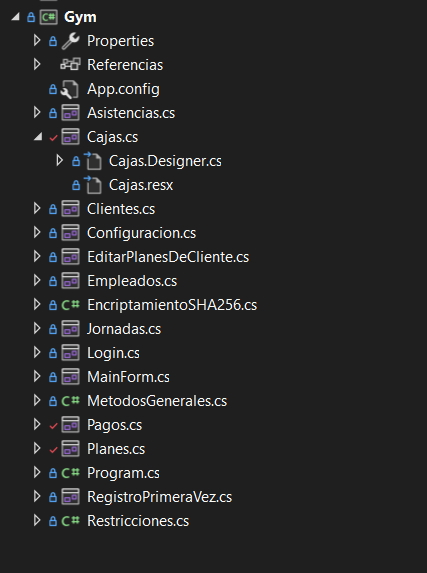
**Gimnasio**

Como se mostrará a continuación, se realizó una solución tipo CRUD en 4 capas o proyectos, Datos, Negocio, Entidades y Presentación, correspondientemente.



**Gym – Capa de presentación.**

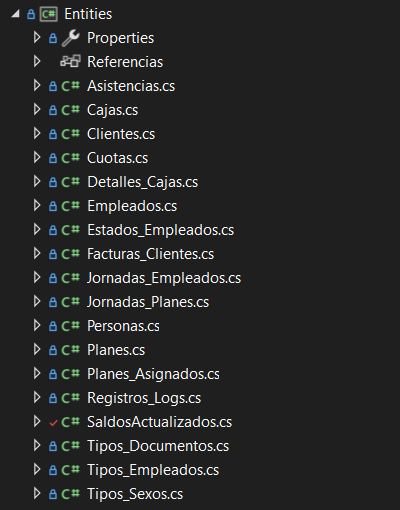
Esta capa se encarga de mostrar todo lo visual que el usuario puede observar (y es la forma en la que accede a toda la solución), y manejar los eventos, acorde a las acciones del usuario. Como un click en un botón, solicitudes ABM, consultar cosas, etc.   
Esta capa tiene relación con la capa de negocio y la capa de entidades. No se relaciona con la capa de Datos, para evitar vulnerabilidades de seguridad, respecto a la información sensible de la base de datos.  
Entonces, como se muestra en la imagen, contiene todos los formularios que el usuario va a poder visualizar, y con los que va a poder interactuar.  
De resto, tiene clases generales (MetodosGenerales, Restricciones... etc), que son comunes a todo el mismo proyecto, a las que se puede acceder desde cualquier formulario.



**Entities – Entidades**

Esta capa, representa cada una de las tablas que hay en la base de datos. Se utiliza para poder asignarle los datos que hay en cada campo de cada tabla que necesitemos consultar, insertar, o modificarle a la misma base de datos.

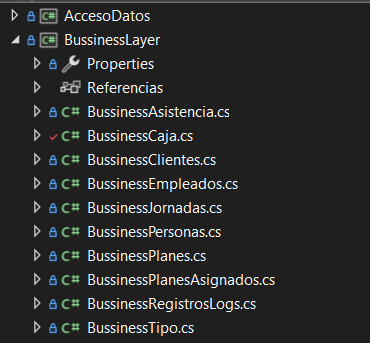
Esta capa no tiene ninguna referencia a relación con otras capas. Ya que las otras capas son las que se relacionan con ella.



**BussinessLayer – Capa de negocio**

Esta capa es la que se encarga de comunicar el front con la capa de datos. Es quien lleva las consultas y activa el funcionamiento de la capa de datos, para que esta última pueda realizar las tareas requeridas.

Esta capa se relaciona con la capa de datos y la capa de entidades.



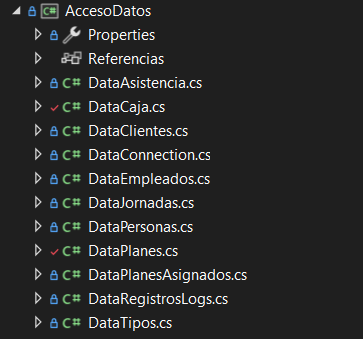
Se crea una clase en esta capa, por cada una de las entidades. Pero se ha decido, en ciertas instancias, optar por tener 1 sola clase para 2 entidades, como por ejemplo las jornadas. Así como también se podría utilizar una clase sola, para las personas y realizar las consultas para los empleados y clientes, pero se optó por separar estas últimas, ya que son muchas líneas, y por cuestiones de ordenamiento y prolijidad para el proyecto.

\*\*\*\*\*\*\*\*\*\*\*\*\*

**DataAccess – Capa de datos.**

Esta última capa es la que realiza las consultas generales a la base de datos. Es la que tiene la conexión a la base de datos, las consultas de todo tipo. Recibe los datos, desde la capa de negocio junto con la capa de entidades, que se requieran escribir o consultar a la base de datos y se encarga de abrir la conexión con la base de datos, enviar la consulta, traer el resultado, y cerrar la conexión a la base de datos.

Esta capa se relaciona solamente con la capa de entidades, ya que la capa de negocio ya se relaciona con ella.



En esta capa, también se crea una clase, por cada una de las entidades que se tiene en la capa de Entidades. Por lo que cada entidad, tiene su acceso a datos. Solo que se creó una clase extra, que es la clase principal, ya que es la que establece la conexión con la base de datos mediante el path correspondiente, y la que tiene los métodos para abrir y cerrar la misma. El resto de las clases, heredan de esta primera clase, para acceder a los métodos de conexión.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Lenguaje utilizado**

Es una solución sencilla, escrita en su 95% en C#, y el 5% restante escrito en SQL, tal como lo menciona GitHub:



A continuación, se va a mostrar el código de todas las clases mostradas, detallando todos los métodos que se utilizaron. La mayoría de los métodos tiene ya comentado el funcionamiento y/o para qué se utiliza.

Los métodos no comentados, es porque realmente no hace falta hacerlo, son fáciles de entender, si sabemos la lógica de programación.  
Ninguna persona que no sepa de programación estaría revisando códigos, por lo que es más simple de esta manera, para no tener que comentar el 100% de cada línea que se escribe, para no sobrecargar visualmente cada una de las clases.