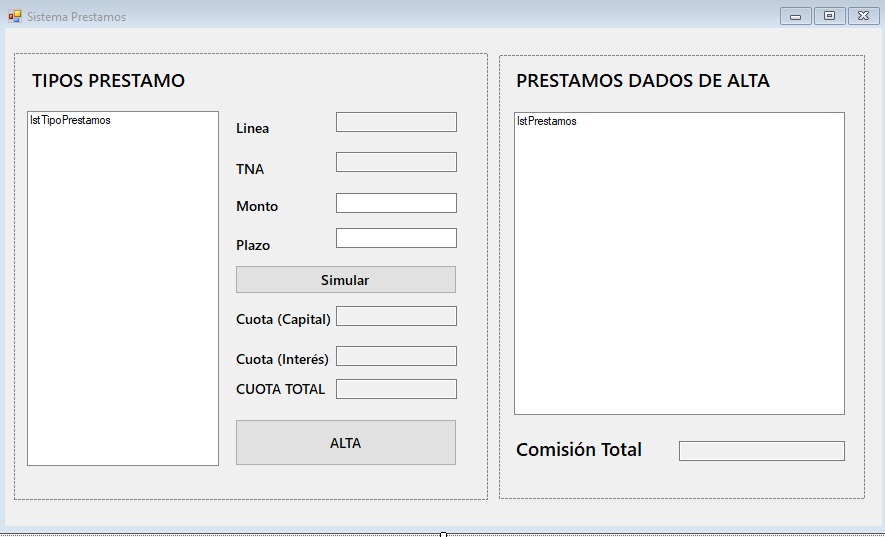
Préstamo

Ejercicio segundo tramo

# Enunciado

Desde la empresa “**La Banca que Banca**” le han solicitado que desarrolle una aplicación para simular y dar de alta los préstamos en base a parámetros de mercado; así como también llevar un registro de todos aquellos que ya fueron ingresados.



Se requiere:

1. Mostrar en pantalla los diferentes tipos de préstamos obtenidos desde el servicio "LineasPrestamo" (GET).
2. Brindar al usuario la posibilidad de seleccionar una línea de préstamo entre las obtenidas en el punto a. (validación)
3. Habilitar al usuario campos de monto y plazo de una línea seleccionada.
4. Mediante el botón simular se evalúa el ingreso de usuario y se calculan las cuotas con las fórmulas dadas en el diseño de la entidad.
5. Mediante el botón de Alta se deben tomar los campos necesarios para realizar el alta del préstamo con el servicio "Prestamo" (POST)
6. Durante la primer carga del formulario y en cada Alta de préstamo, se debe actualizar la lista lstPrestamos con lo obtenido en el servicio "Prestamo" (GET)
7. Cada vez que se carga lstPrestamos se debe calcular la comisión (ver entidad operador)

\* Los campos grisados son los que el usuario NO puede editar.

\*\* Se valorará el uso de archivo de configuración para los valores estáticos (url, registro de alumno, etc)

\*\*\* Se valorará la presentación del registro en la lista. (Formato ejemplo: **id**) **Días**  dias - ARS **CapitalInicial** (interés **Interés**) – tipo o (**desc del tipo plazo fijo)** - )

# Objetivos mínimos

1. Arquitectura en capas con responsabilidades delimitadas de cada capa
2. Validación de inputs
3. Serialización-Deserialización
4. Interfaz de usuario en forms
5. Utilizar los conceptos vistos en POO
6. Desarrollar los algoritmos requeridos

# Referencia

Documentación y pruebas de API: <https://cai-api.azurewebsites.net/swagger/index.html>

## Entidades

### Operador

+ Prestamos: List<Prestamo>

+ Comision: double

+ PorcentajeComision: double

Parámetros: El porcentaje de comisión es el 15% del interés total para cada préstamo

### ResultadoTransaccion

+ isOk: bool

+ error: string

+ id: int

### TipoPrestamo

+ Linea: string

+ TNA: double

+ id: int

### Prestamo

+ Linea: string

+ TNA: double

+ Plazo: int

+ Monto: double

+ Usuario: string

+ id: int

Nota: En el préstamo se podría aplicar agregación y reemplazar “Linea” y “TNA” por el identificador de TipoPrestamo, pero se toma esta aproximación de “snapshot” para simplificar la implementación.

#### Propiedades de la entidad

+ CuotaCapital: Monto/Plazo

+ CuotaInteres: CuotaCapital \* (TNA/12/100)

+ Cuota: CuotaCapital + CuotaInteres

## Servicios

### LineasPrestamo

**GET**: <https://cai-api.azurewebsites.net/api/v1/prestamotipo>

**RETURN**: List<TipoPrestamo>

### Prestamo

**GET**: [https://cai-api.azurewebsites.net/api/v1/prestamo/{registro](https://cai-api.azurewebsites.net/api/v1/prestamo/%7Bregistro)}

**RETURN**: List<Prestamo>

**POST**: <https://cai-api.azurewebsites.net/api/v1/prestamo/>

**INPUT**: Linea, Plazo, Monto, CuotaTotal

**RETURN**: ResultadoTransaccion

Notas:

* *Tanto id como idCliente se debe enviar en 0*
* *Para enviar los double usar* ***.ToString("0.00")***
* *Respetar los case. Solo id y idCliente comienzan con minúscula.*