**Análisis y diseño**

### Clase UserInfo

* Atributos:
  + usuario (String): Nombre del usuario.
  + id (int): Identificación del usuario.
  + publicKey (String): Clave pública del usuario.
  + monto (double): Monto de la transacción.
  + cuotas (int): Cuotas para el pago (opcional).
  + numeroDeTarjeta (long): Número de tarjeta de crédito.
  + fechaDeVencimiento (int): Fecha de vencimiento de la tarjeta.
  + codigoCVV (int): Código CVV de la tarjeta.

### Interfaz Interfaz

* Métodos:
  + generatePaymentFile(ArrayList<UserInfo> users, String filename): Método para generar archivos de pago, requiere una lista de usuarios y un nombre de archivo como parámetros.

### Clase American (implementa Interfaz)

* Métodos:
  + generatePaymentFile(ArrayList<UserInfo> users, String filename): Genera un archivo de pago en formato CSV para tarjetas American Express. Calcula el servicio, agrega porcentaje según las cuotas y genera un código hash MD5.

### Clase Mastercard (implementa Interfaz)

* Métodos:
  + generatePaymentFile(ArrayList<UserInfo> users, String filename): Genera un archivo de pago en formato JSON para tarjetas Mastercard. Convierte el monto a dólares, encripta el número de tarjeta y calcula el hash MD5 para el contenido.

### Clase Visa (implementa Interfaz)

* Métodos:
  + generatePaymentFile(ArrayList<UserInfo> users, String filename): Genera un archivo de pago en formato XML para tarjetas Visa. Calcula el monto con el servicio, transforma la información del usuario a elementos XML y calcula el hash MD5.

### Clase ReadFiles

* Métodos:
  + readJSON(String filename): Lee un archivo JSON y devuelve una lista de objetos UserInfo.

### Clase Menu

* Métodos:
  + menu(): Interactúa con el usuario, procesa los archivos y llama a los métodos para generar los archivos de pago.

### Clase App

* Métodos:
  + main(String[] args): Punto de entrada del programa. Inicia la aplicación llamando al menú.