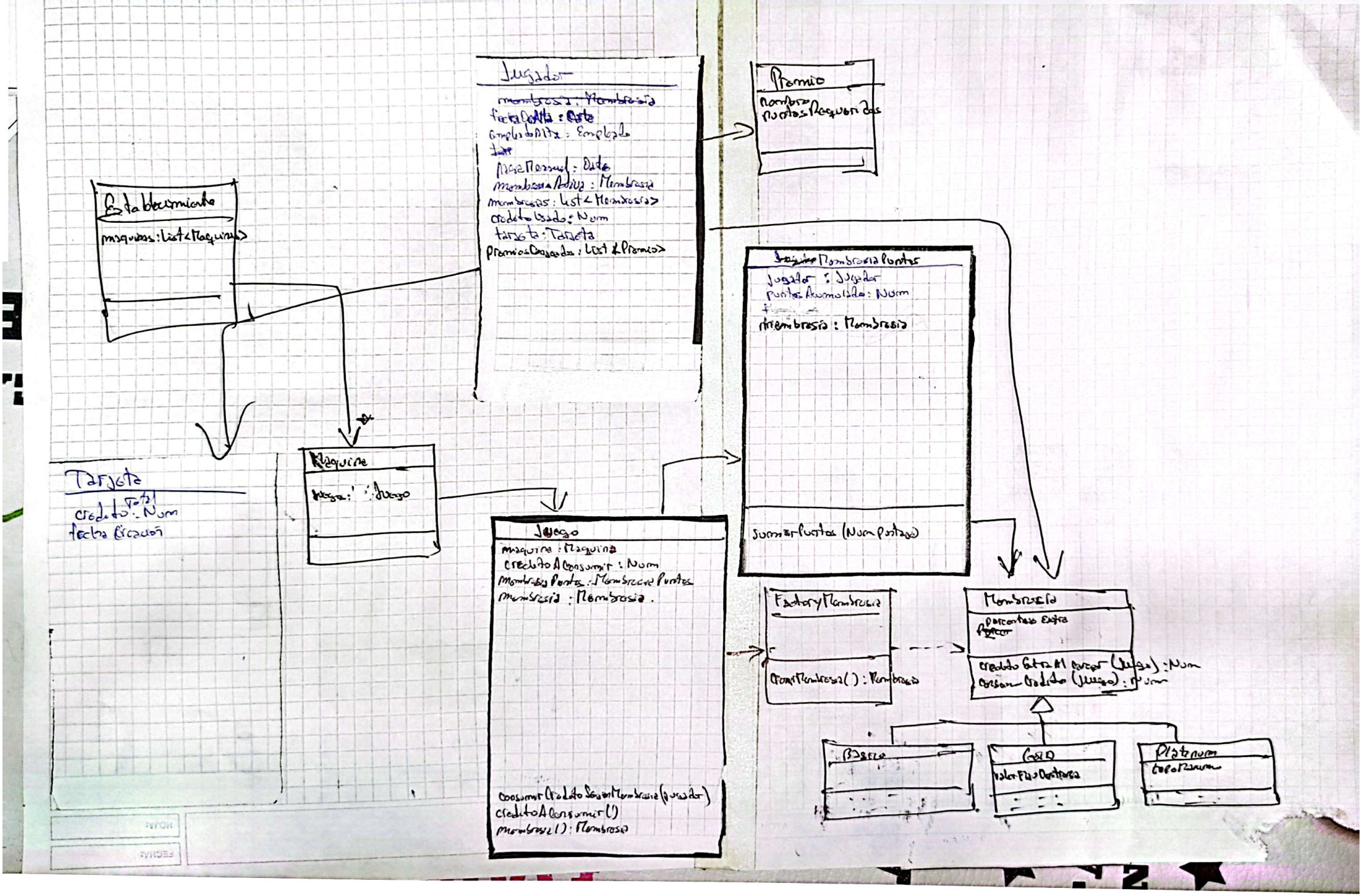
Jugador
Premio
Establecimiento
Tarjeta
Maquina
Juego
MembresiaPuntos
FactoryMembresia
Membresia
Bronce
Gold
Platinum

HOJA N°

FECHA

3) // Recargar cierta cantidad de créditos

#Jugador >> consumirCreditoSegunMembresia (Juego juego ~~Juego~~

3) // recargar cierta cantidad de credito

3.1)

```
#Juego >> consumirCreditoSegunMembresia (Jugador jugador) {
    if (jugador.membresia is null)
        jugador.membresia = FactoryMembresia.crearMembresia (jugador.TipoMembresia)

Var consumo = jugador.membresia.consumoCredito (self);
jugador.descontarCredito (consumo);
return consumo;
}
```

TipoMembresia
- BasicoTipo
- PlatinumTipo
- GoldTipo

```
#FactoryMembresia >> crearMembresia (TipoMembresia tipo) {
    Membresia membresia = null;
    Switch (tipo):
        caso BasicoTipo : membresia = Basic() obtenerInstancia(); break;
        caso GoldTipo : membresia = Gold() obtenerInstancia(); break;
        caso PlatinumTipo : membresia = Platinum() obtenerInstancia(); break;
        default new Membresia Excepcion("no existe este tipo");
    return membresia;
}
```

NOTA

HOJA N°

FECHA

3) // Recargar cierta cantidad de creditos

#Jugador >> consumirCreditoSegunMembresia (Juego juego ~~Jugador~~)

3) // recargar cierta cantidad de credito

3.1) #Juego >> consumirCreditoSegunMembresia (Jugador jugador) {

```
if (jugador.membresia is null)
    jugador.membresia = FactoryMembresia.crearMembresia(jugador.TipoMembresia)

Var consumo = jugador.membresia.consumoCredito(self);
jugador.descontarCredito(consumo);
return consumo;
}
```

TipoMembresia
- BasicoTipo
- PlatinumTipo
- GoldTipo

#FactoryMembresia >> crearMembresia (TipoMembresia tipo) {

```
Membresia membresia = null;

Switch (tipo):
    case BasicoTipo  : membresia = Basico(); break;
    case GoldTipo    : membresia = Gold(); break;
    case PlatinumTipo: membresia = Platinum(); break;
    default new MembresiaException("no existe este tipo");

return membresia;
}
```

NOTA

3.2) // Registrar una partida que llega de una máquina
// acá el controller procesa el JSON que recibe de la maquinola

ControllerPartida >> recibirPartida (... PartidaJSON ... datos ...) { // puede ser un DTO

Jugador jugador = serviceJugador.buscarPorId(datos.jugador_id)

Juego juego = serviceJuego.buscarPorId(datos.juego_id)

Maquina maquina = serviceMaquina.buscarPorId(datos.maquina_id)

[redacted]

jugador.procesarCredito(juego)
jugador.procesarPuntos(datos.puntaje)
jugador.procesarTiempoDeJuego(datos.duracion_segundos)

}

Jugador >> procesarCredito(Juego juego) {

var credito = ... self.membresia.consumoCredito(juego)

self.credito -= credito;

}

Jugador >> procesarPuntos(Num puntaje) {

[redacted]

self.membresia.sumarPuntos(puntaje);

}

MembresiaJugador >> sumarPuntos(Num puntaje) {

self.puntos += puntaje;

}

Basico >> consumoCredito(Juego juego) {

return self.porcentajeConsumo.juego.creditoAConsumir()

}

Gold >> consumoCredito(Juego juego) {

var credito = 0;
if (juego.TipoJuego == DestrezaTipo) credito += juego.creditoAConsumir() - self.valorFijoDestreza;

return credito;

}

Platinum >> consumoCredito(Juego juego) {

return juego.creditoAConsumir() * self.porcentajeDescuento

}

3.3) // Canjear un premio sea en un establecimiento o por internet

```
#Jugador => canjear (premio) {
        if (self.puntosAcumulados > premio.puntos)
            self.puntosAcumulados -= premio.puntos;
        }
        else new CanjeException ("No se puede canjear")
    }

#Jugador => canjearConLogistica (premio, datosDeEnvioApi) {
        self.canjear(premio);
        ServiceLogistica.enviar (premio, datosDeEnvioApi);
    }
```