# Pattern Matching

*TADP - 2018 C2 - TP Metaprogramación*

## Entrega Individual

El objetivo de esta entrega es extender el trabajo práctico grupal para agregar una nueva funcionalidad. Este agregado debe hacerse utilizando las herramientas vistas en clase, teniendo especial cuidado de no romper la funcionalidad provista en la entrega anterior.

### Bien, más combinators!

En la entrega grupal del trabajo práctico, se pedía que desarrollaramos un framework que nos permitiera hacer pattern matching. Sólo pedimos que se implementaran los matchers **val, list, duck, type,** y también permitir el matcheo de **variables**.

#### Combinator if

Se pide que estos combinators funcionen sólo sobre **variables**, con el propósito de poder agregarles una condición, ya que hasta ahora las variables siempre devolvían true porque eran el matcher de **identidad**.
Los bloques de condiciones se evalúan sobre el valor al que se ligaría cada variable.

```
(:a_string.if do length < 10 end).call("this is not a
string") # => false

(:a_string.if do length < 10 end).call("wololo") # => true

(:a_number.if do even? end).call(42) # => true

(:a_number.if do even? end).call(23) # => false

matches([1,2,3]) do
with((list([:uno.if do odd? end,:dos,:tres])) {uno + dos +
tres}
otherwise { 1 }
end #=> 6
```

Además, se pide que si la variable no llega a entender algún mensaje, permitir que se propague el error correspondiente, en vez de devolver algún booleano.

```
(:a_number.if do length? end).call(23) # => NoMethodError

(:a_number.if do even? end).call(Guerrero.new) # =>
NoMethodError
```

Notar que esto debería seguir funcionando:

```
matches?(10) do
with(:a_number.if do odd? end){a_number}
with(:a_number.if do even? end){a_number * 2}
with(:a_number.if do nil? end){a_number / 2}
end # => 20
```