

dCode

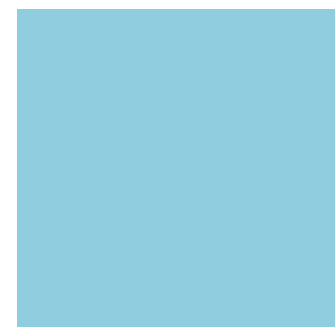
---


nielsen indices

¡Hola!



Natalia Castellanos / Santiago Martínez





## ¿Cuál fue el reto que nos planteamos?

- Necesitamos un conocimiento superior del cliente.
- No necesariamente lo vamos a obtener de “exprimir” más la data estructurada.
- Cantidades de información que no se analiza de manera masiva e industrializada.
- Textos amplios, voz de clientes, preguntas abiertas.
- Y allí hay respuestas del cliente que necesitamos y que aprovecharla genera Insights nuevos.



Según IBM, los datos no estructurados son el 80% de la información disponible

Todo lo que no sea accionable en una base de datos de "filas y columnas", es información no estructurada.



dCode

Identificamos que debíamos lograr trabajar intensivamente con audios, textos, chats y posts de redes sociales.



Voz

Una fuente fantástica de conocimiento  
del cliente

dCode

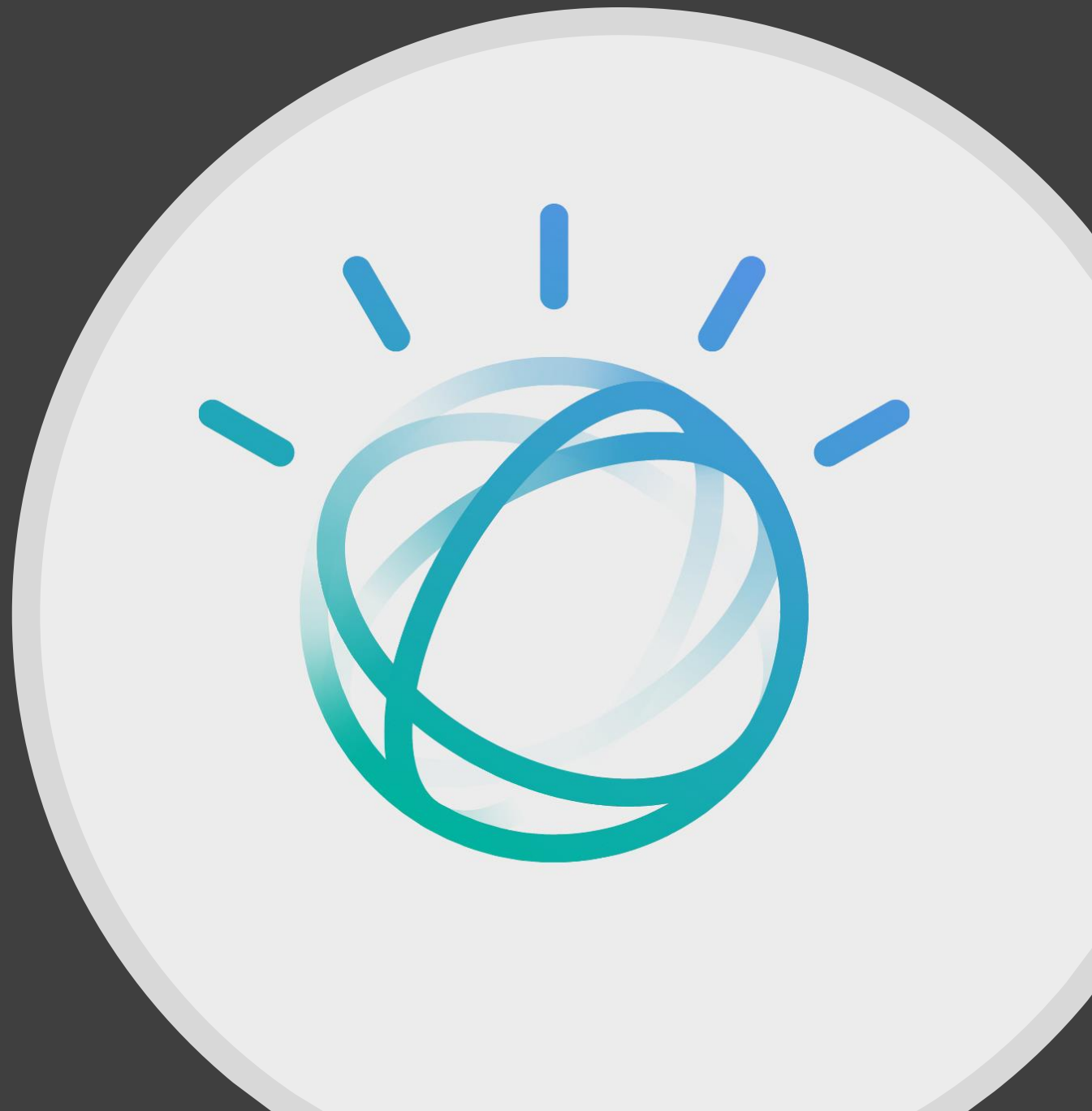
# Textos

Redes sociales, chats (bots),  
campos abiertos en encuestas y  
sugerencias de cliente.

```
extract_number_and_incr (destination, source) int
*destination; unsigned char **source; { extract_number_and_incr (destination, *source); *source += 2; } #ifndef EXTRACT_MACROS #undef EXTRACT_NUMBER_AND_INCR #define EXTRACT_NUMBER_AND_INCR(dest, src) \ extract_number_and_incr (&dest, &src) #endif /* not EXTRACT_MACROS */ #endif /* DEBUG */ /* If DEBUG is defined, Regex prints many voluminous messages about what it is doing (if the variable `debug' is nonzero). If linked with the main program in `iregex.c', you can enter patterns and strings interactively. And if linked with the main program in `main.c' and the other test files, you can run the already-written tests. */ #ifdef DEBUG /* We use standard I/O for debugging. */ #include <stdio.h> /* It is useful to test things that ``must'' be true when debugging. */ #include <assert.h> static int debug = 0; #define DEBUG_STATEMENT(e) e #define DEBUG_PRINT1(x) if (debug) printf (x) #define DEBUG_PRINT2(x1, x2) if (debug) printf (x1, x2) #define DEBUG_PRINT3(x1, x2, x3) if (debug) printf (x1, x2, x3) #define DEBUG_PRINT4(x1, x2, x3, x4) if (debug) printf (x1, x2, x3, x4) #define DEBUG_PRINT_COMPILED_PATTERN(p, s, e) \ if (debug) print_partial_compiled_pattern (s, e) #define DEBUG_PRINT_DOUBLE_STRING(w, s1, sz1, s2, sz2) \ if (debug) print_double_string (w, s1, sz1, s2, sz2) extern void printchar(); /* Print the fastmap in human-readable form. */ void print_fastmap (fastmap) char *fastmap; { unsigned was_a_range = 0; unsigned i = 0; while (i < (1 << BYTEWIDTH)) { if (fastmap[i++] { was_a_range = 0; printchar (i - 1); while (i < (1 << BYTEWIDTH) && fastmap[i]) { was_a_range = 1; i++; } if (was_a_range) { printf ("-"); printchar (i - 1); } } putchar ('\n'); } /* Print a compiled pattern string in human-readable form, starting at the START pointer into it and ending just before the pointer END. */ void print_partial_compiled_pattern (start, end) unsigned char *start; unsigned char *end; { int mcnt, mcnt2; unsigned char *p = start; unsigned char *pend = end; if (start == NULL) { printf("(null)\n"); return; } /* Loop over pattern commands. */ while (p < pend) { switch ((re_opcode_t) *p++) { case no_op: printf("/no_op"); break; case exactn: mcnt = *p++; printf("/exactn/%d", mcnt); do { putchar ('/'); printchar (*p++); } while (--mcnt); break; case start_memory: mcnt = *p++; printf("/start_memory/%d/%d", mcnt, *p++); break; case stop_memory: mcnt = *p++; printf("/stop_memory/%d/%d", mcnt, *p++); break; case duplicate: printf("/duplicate/%d", *p++); break; case anychar: printf("/anychar"); break; case charset: case charset_not: { register int c; printf("/charset%s", (re_opcode_t) *(p - 1) == charset_not ? "_not" : ""); assert (p + *p < pend); for (c = 0; c < *p; c++) { unsigned bit; unsigned char map_byte = p[1 + c]; putchar ('/'); for (bit = 0; bit < BYTEWIDTH; bit++) if (map_byte & (1 << bit)) printchar (c * BYTEWIDTH + bit); } p += 1 + *p; break; } case begline: printf("/begline"); break; case endlime: printf("/endlime"); break; case on_failure_jump: extract_number_and_incr (&mcnt, &p); printf("/on_failure_jump/0/%d", mcnt); break; case on_failure_keep_string_jump: extract_number_and_incr (&mcnt, &p); printf("/on_failure_keep_string_jump/0/%d", mcnt); break; case dummy_failure_jump: extract_number_and_incr (&mcnt, &p); printf("/dummy_failure_jump/0/%d", mcnt); break; case push_dummy_failure: printf("/push_dummy_failure"); break; case maybe_pop_jump: extract_number_and_incr (&mcnt, &p); printf("/maybe_pop_jump/0/%d", mcnt); break; case pop_failure_jump: extract_number_and_incr (&mcnt, &p); printf("/pop_failure_jump/0/%d", mcnt); break; case jump_past_alt: extract_number_and_incr (&mcnt, &p); printf("/-
```

Entendimos que el mejor aliado es IBM®  
Watson®

Para lograr una aplicación  
concreta enfocada en el  
entendimiento del cliente  
con tecnología cognitiva





Personalidad, emociones y sentimientos.

¿Cuál es la aplicación al negocio?

¿Para qué?





# La personalidad como rector de las decisiones

Si la conocemos somos más acertados en nuestras acciones de marketing



# Medir la Experiencia, sin preguntar por ella

Porque no es una respuesta, es una  
combinación de emociones y sentimientos

dCode

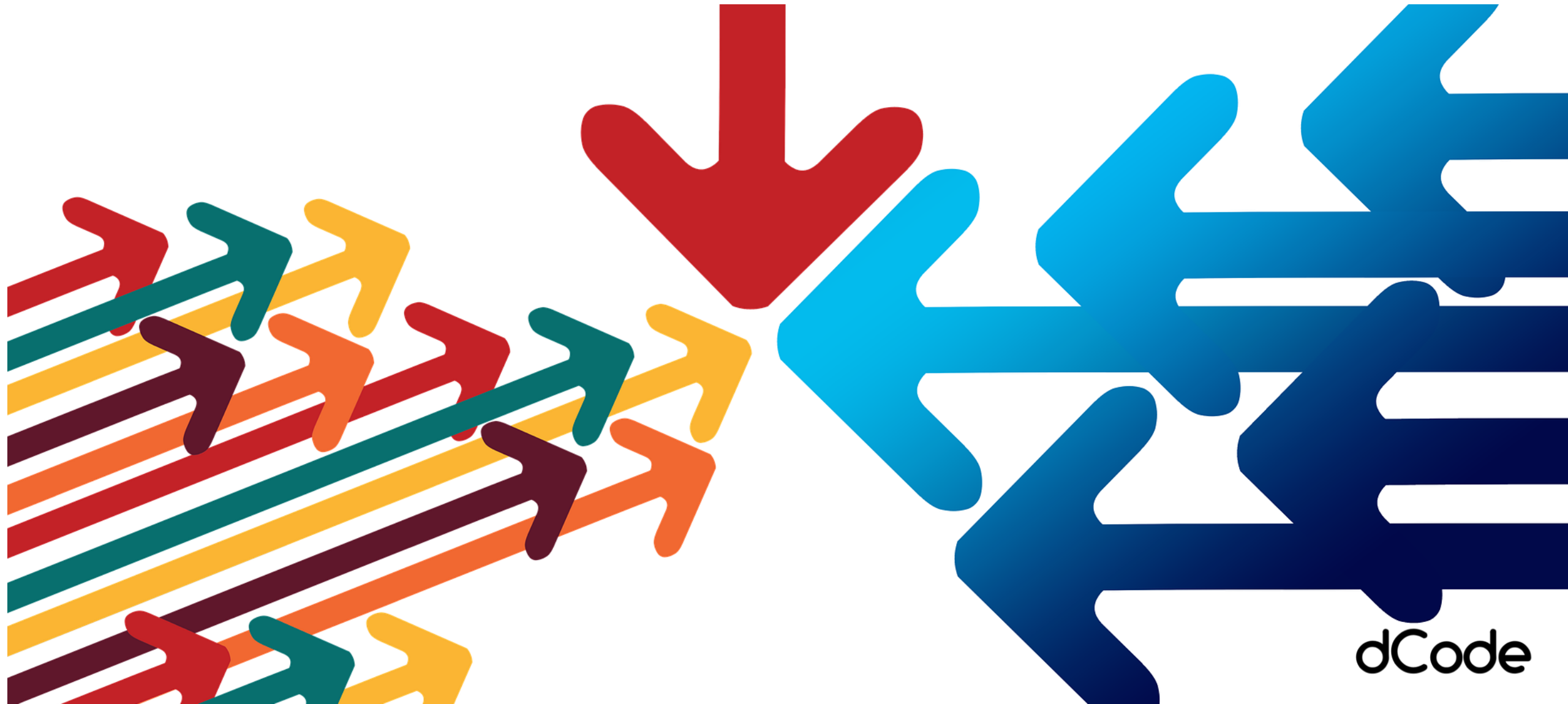
An abstract, colorful fractal background featuring swirling patterns of red, orange, yellow, green, and blue, resembling a complex, organic structure. The colors transition from warm reds and oranges on the left to cooler blues and greens on the right, with bright yellow and white highlights at the center of the swirls.

# Segmentaciones

Basadas en las personas  
como individuos

dCode

Conectados con las bases de datos del cliente para hacerlos accionables.

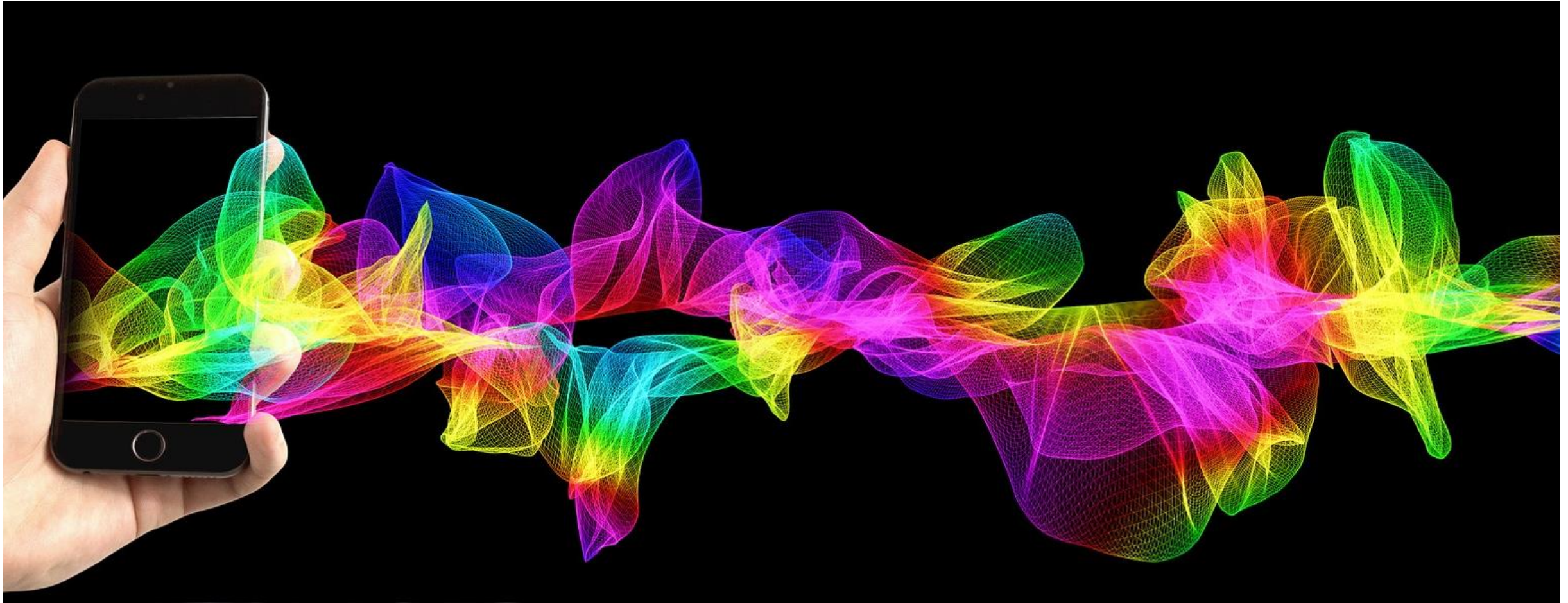




# Audiencias

Para comunicarnos de  
manera más acertada

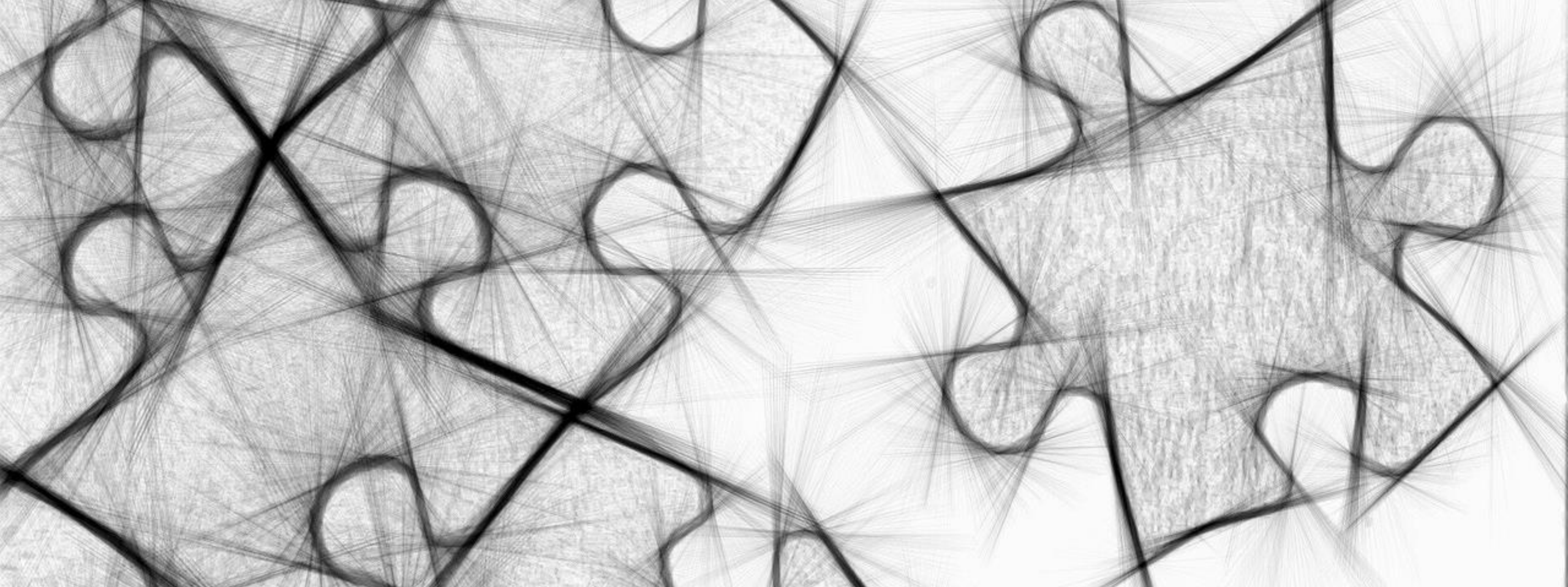
dCode



¿Digitales? ¿Leales?  
¿Exigentes?  
¿Arriesgados?

# Perfiles

dCode



# Necesitamos juntar habilidades

Estamos en el siglo de la  
cooperación y de la  
innovación abierta

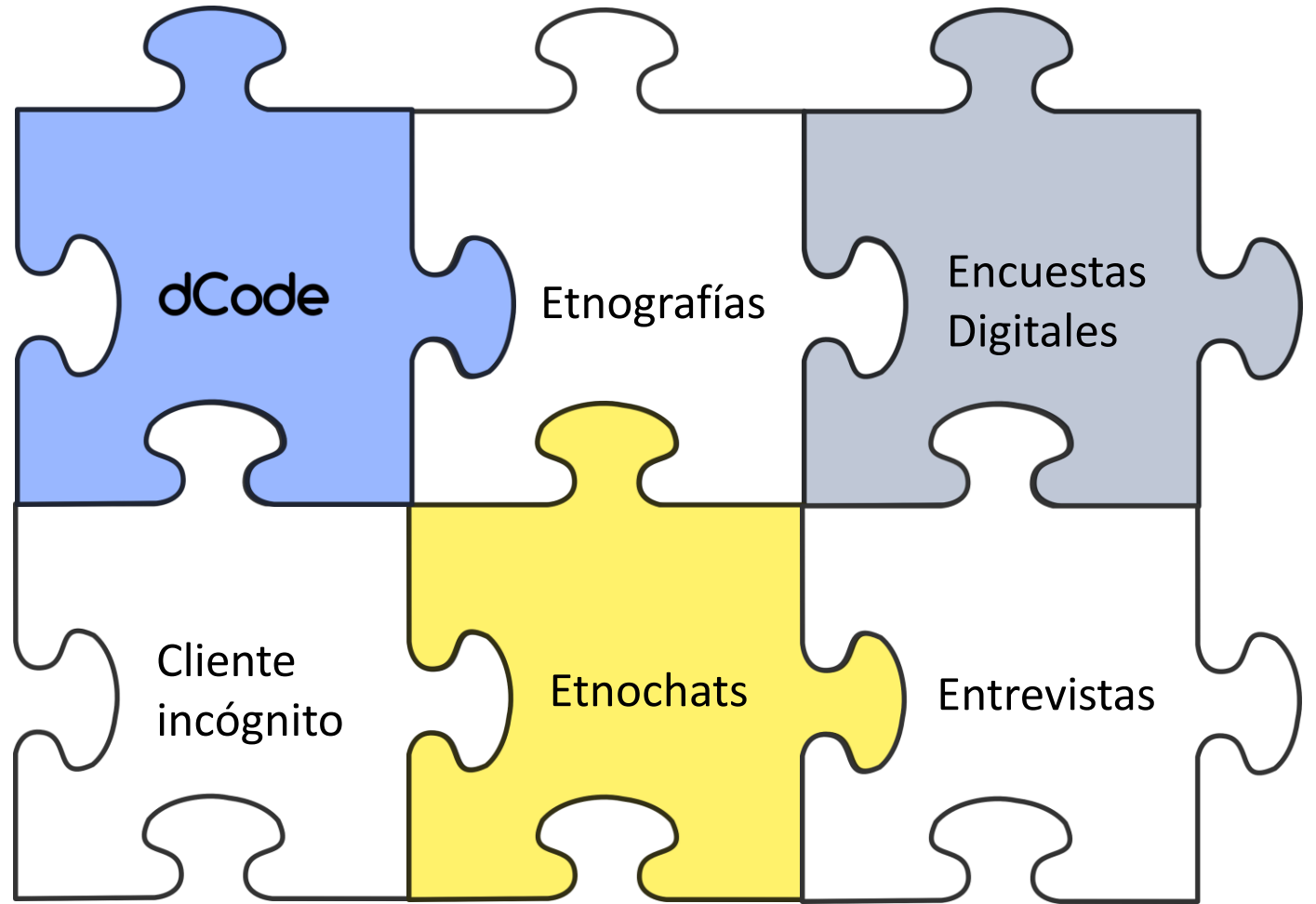
dCode

(Nielsen + indices)^Watson =  
dCode

---

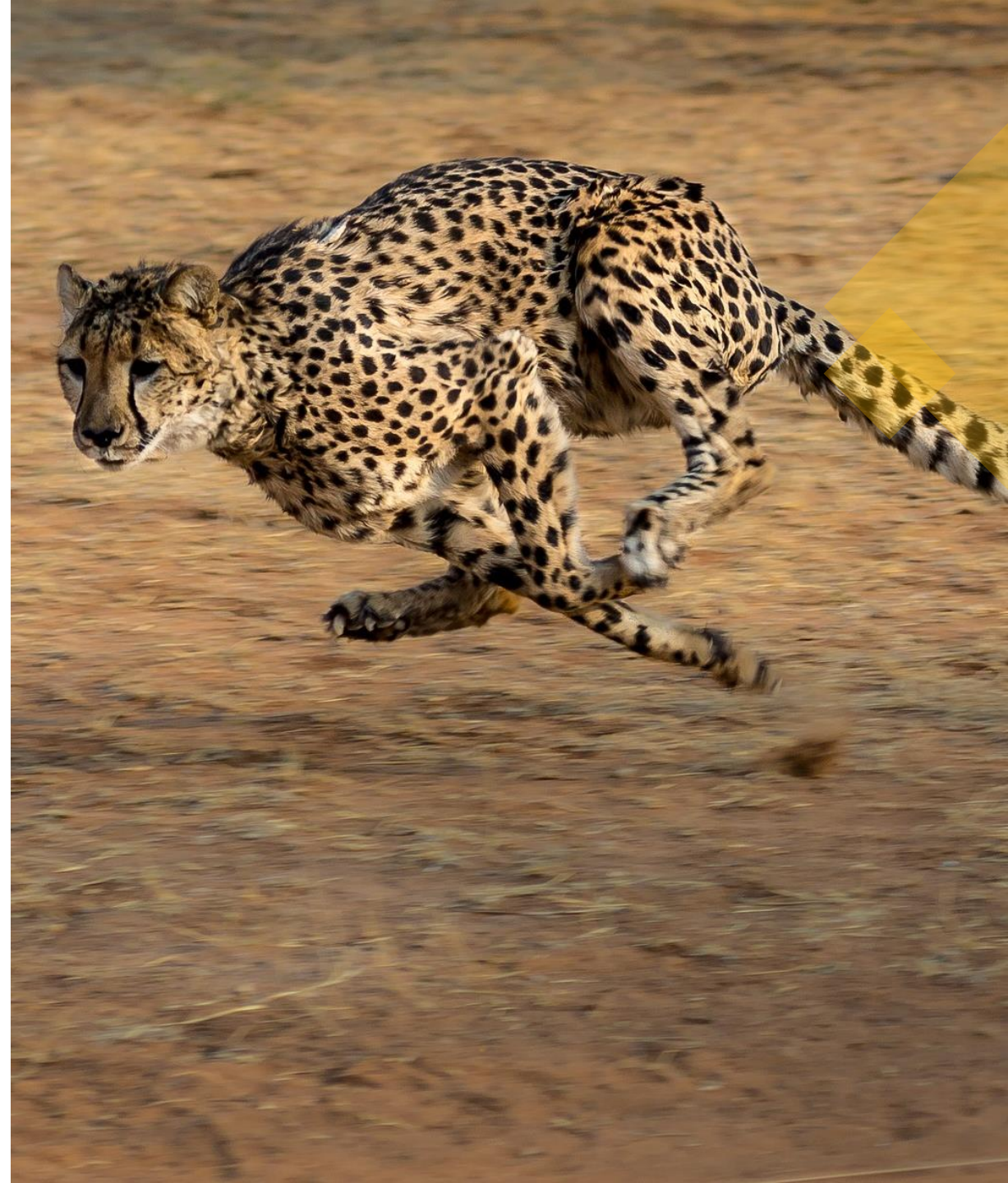
Es nuestra ecuación


**Adaptamos** de manera flexible las metodologías y herramientas, para obtener resultados disruptivos:



# Caso de éxito

- Grupo éxito, Colombia.
- Parte de la necesidad de ajustar las evaluaciones de servicio, enfocándose en los clientes y no en sus transacciones.
- Soluciones tradicionales los hacían partir de supuestos para poder avanzar con un modelo de segmentación que permitiera ajustar el modelo de contacto.
- **Se analizaron +8.000 minutos de audio de clientes en una semana.**
- dCode llega con una solución disruptiva que parte del cliente y genera cinco segmentos cognitivos que permitieron:
  - Saber qué preguntar, a quién, con qué lenguaje y con qué tono.
  - Avanzar en un modelo de comunicación en donde, desde los elementos cognitivos, se hacía más eficiente cada contacto con el cliente.
  - Inferir desde los casos realizados, a una primer muestra de clientes, el perfil cognitivo más cercano del universo de clientes trabajado.
- El objetivo hoy es conectar el 100% de la base de datos de clientes a los segmentos encontrados, para lograr accionabilidad en el total de clientes.





Porque somos personas  
más que clientes

El negocio del futuro debe estar realmente  
centrado en el cliente como persona, no solamente  
como transacciones.

dCode

# ¡Gracias!

**Contactos:**

Santiago Martínez Vela: [smartinez@indices.com.co](mailto:smartinez@indices.com.co)

Natalia Castellanos: [maria.castellanosotero@nielsen.com](mailto:maria.castellanosotero@nielsen.com)

dCode

---

nielsen indices