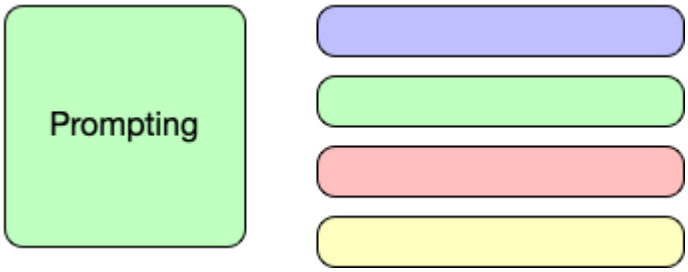
Hoy por hoy me encuentro con que muchas personas me comentan. Que la IA lo hace todo y que no tenemos que tener conocimiento. Sin embargo hablando con clientes … me he encontrado que algunas personas ya han tenido los primeros sustos …. con perfiles que usaron la IA …. y abusaron de la IA?.

Abusar de la IA?

Estamos abusando de la IA? . Es una pregunta muy delicada porque si decimos que si… tendremos mucha gente que se nos echará en cima y dirá que hay q abrazar la innovación y que vivimos en el pasado . Si decimos lo contrario y decimos que no estamos abusando de la IA nos encontraremos con otro grupo de personas que nos dirá que acabaremos en el infierno del desarrollo.

La IA y el conomiento

Para manejarse con la IA se necesita conocimiento . Es una cosa curiosa porque en principio la IA nos vende que el conocimiento deja de tener valor ya que ella nos lo da la solución a todo. Sin embargo no es así de sencillo …. la IA nos da una “solución a todo” no “nuestra solución” , “no la solución final” sino “una solucion” que estadísticamente es la ganadora. Vamos a ver un ejemplo



La clase Factura en Lombok

Si yo le pido a un motor de IA la clase Factura con Lombok es probable que nos devuelve esto

```
package com.arquitecturajava.modelo;

import lombok.Data;

import java.math.BigDecimal;
import java.time.LocalDate;

@Data
public class Factura {

    private String id;
    private String numero;
    private LocalDate fechaEmision;
    private String cliente;
    private BigDecimal importe;
    private boolean pagada;
}
```

Es esta clase correcta?. Uhhm depende … pero me lo ha devuelto claude… así que en principio es una solución. Si se lo pido a chatgpt

```
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.math.BigDecimal;
import java.time.LocalDate;

@Data
@Builder
@NoArgsConstructor
```

```
@AllArgsConstructor
public class Factura {

    private Long id;
    private String numero;
    private LocalDate fecha;
    private String cliente;
    private String concepto;
    private BigDecimal importe;
    private BigDecimal iva;
    private BigDecimal total;
    private boolean pagada;
}
```

Vemos que la solución es un poco complementaria… son dos enfoque con dos motores de IA pero todas van a tener problemas con @Data ya que si son clases de negocio pueden tener problemas a futuro con los toString, equals y hashcode. Si cambio el prompt y pongo "genera una clase de negocio factura con lombok"

```
package com.arquitecturajava.negocio;

import lombok.Getter;
import lombok.Setter;
import lombok.ToString;

import java.math.BigDecimal;
import java.time.LocalDate;

@Getter
@Setter
@ToString
public class Factura {
```

```
    private String numero;
    private String cliente;
    private LocalDate fechaEmision;
    private BigDecimal baseImponible;
    private BigDecimal tipoIva = new BigDecimal("21.00");
    private boolean pagada;

    public BigDecimal getTotal() {
        BigDecimal cuotaIva = baseImponible
                .multiply(tipoIva)
                .divide(new BigDecimal("100"));
        return baseImponible.add(cuotaIva);
    }

    public void marcarComoPagada() {
        this.pagada = true;
    }
}
```

La clase mejora mucho ya que no incluye los constructores , ni el equals y hashcode que añade lombok por defecto con @Data y que en muchos casos dan quebradores de cabeza en clases de negocio. Solo hemos cambiado una palabra de “clase Factura” a “clase de negocio Factura” y el resultado ha cambiado de forma significativa.

El conocimiento importa más que nunca

Ahora nuestra función no es “codificar” sino es “decidir ” y “orientar” las herramientas aportando un valor .En este caso necesitaremos más conocimiento por parte de todos y no menos. Eso nos ayudará a producir más código y de más calidad.

- Java Lombok , clases y productividad
- Prompt Engineering patrones

- Lombok en Java para ahorrar tokens y codigo
- Curso Spring AI : Inteligencia artificial en Java
- Java @Data y preprocesado de JavaBeans