

Piloter l'agent : La Qualité à l'Ère de l'IA

De l'Artisanat du Code à
l'Architecture d'Agents

Sommaire

L'ÉVOLUTION

De l'Assistant à l'Agent

LE DÉFI

Le "Vibe Coding"

L'ANALYSE

Le Spectre des Outils

LA SOLUTION

Méthodologies / Outils
de Pilotage

DEMONSTRATION

Démos & Live-Coding

CONCLUSION & QCM

De l'Artisanat à l'Agent

1.

Code "de base"

Le dev fait 100%

2.

Assistant IA

L'IA suggère. Le
dev reste pilote

3.

Agent IA

L'IA exécute. Elle
planifie, utilise des
outils (code, web)
et devient
autonome

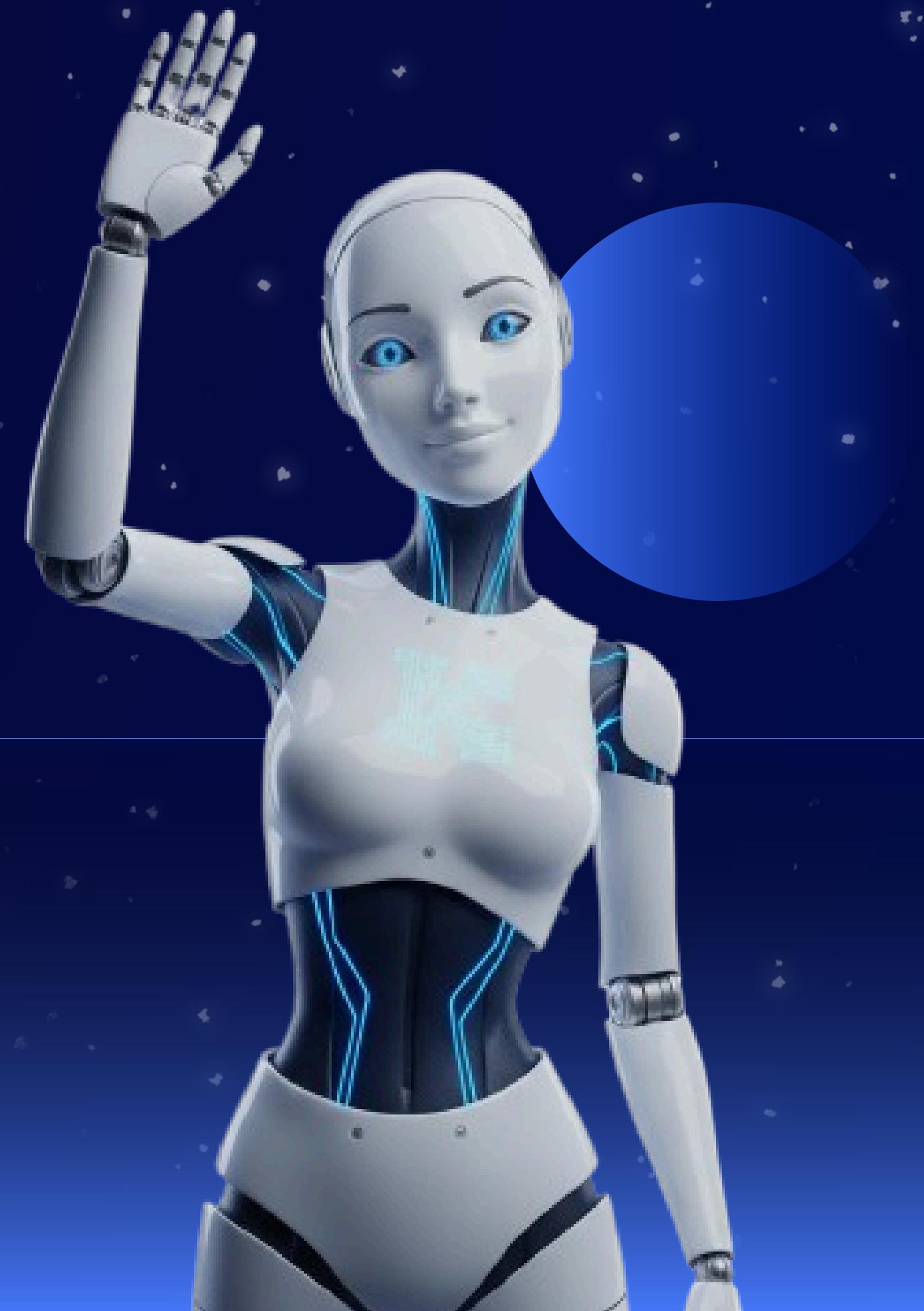
Le "Vibe Coding"

La Vitesse sans Contrôle

VIBE CODING

Une nouvelle
puissance qui emmène
un nouveau risque

RAGT





```
public class GestionPanier {  
    public static List<Produit> ajouterAuPanier(  
        List<Produit> panier, Produit produit) {  
  
        panier. add(produit) ;  
        return panier;  
    }  
}
```

```
public class Produit {  
    private String nom;  
    private double prix;  
  
    public Produit(String nom, double prix) {  
        this.nom = nom;  
        this.prix = prix;  
    }  
  
    public String getNom() {  
        return nom;  
    }  
  
    public double getPrix() {  
        return prix ;  
    }  
}
```





```
public class GestionPanier {  
    public static List<Produit> ajouterAuPanier(  
        List<Produit> panier, Produit produit) {  
  
        panier. add(produit) ;  
        return panier;  
    }  
}
```

```
public class Produit {  
    private String nom;  
    private double prix;  
  
    public Produit(String nom, double prix) {  
        this.nom = nom;  
        this.prix = prix;  
    }  
  
    public String getNom() {  
        return nom;  
    }  
  
    public double getPrix() {  
        return prix ;  
    }  
}
```



Le Code "Plausible mais Faux"

- ✗ Problème Métier
- ✗ Problème de Qualité (POO)
- ✗ Problème de Test

Les problèmes de qualité

Les différents “code smells” :

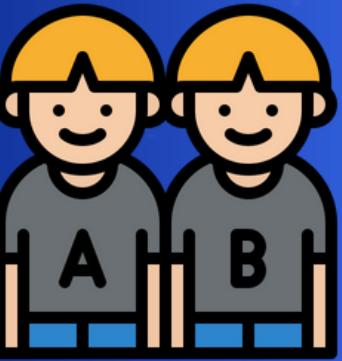
- Le code spaghetti



- Les magic numbers



- Duplication de code



- Le code mort



Comparaison

Approche	Outils	Qualité
Code "de base"	Le cerveau	Dépend du développeur
Assistant IA	Github Copilot	Le développeur reste le garant
"Vibe Coding"	GPT, Claude	Mauvaise
Agent IA	CrewAI, n8n	Potentiellement maximale

Le Coût de la Non-Qualité

Vitesse d'Écriture \neq l'Efficacité
court terme \rightarrow 100 fois plus

test unitaire :

- code sans erreur
- une erreur en production



Les Piliers de la Qualité

L'Architecture d'Agents :

- Agent Codeur
- Agent de test
- Agent Qualité



Agent avec outils :

- sonarcube
- autoanalyse



Agent Codeur



```
public class Produit {  
    private String nom;  
    private double prix;  
    public Produit(String nom, double prix) {  
        this.nom = nom; this.prix = prix;  
    }  
    public String getNom() {  
        return nom;  
    }  
    public double getPrix() {  
        return prix;  
    }  
}
```



```
public class GestionPanier {  
  
    public static List<Produit> ajouterAuPanier(  
        List<Produit> panier, Produit produit) {  
  
        panier.add(produit);  
        return panier;  
    }  
}
```

Agent Testeur

```
● ● ●

class PanierTest {

    private Panier panier;
    private Produit pomme;
    private Produit orange;

    @BeforeEach
    void setUp() {
        // Initialisation avant chaque test
        panier = new Panier();
        pomme = new Produit("P01", "Pomme", 1.50);
        orange = new Produit("002", "Orange", 2.00);
    }

    @Test
    void testGestionQuantite_ProduitIdentique() {
        // C'EST LE TEST CLÉ
        // On ajoute le MÊME produit en deux fois
        panier.ajouterProduit(pomme, 2);
        panier.ajouterProduit(pomme, 3);

        assertEquals(1, panier.getItems().size(), "Le panier ne devrait avoir qu'1 ligne");
        assertEquals(5, panier.getItems().get(pomme), "Les quantités doivent fusionner (2+3=5)");
    }
}
```

Agent qualité

```
public class Produit {  
  
    private final String reference;  
    private final String nom;  
    private final double prix;  
  
    public Produit(String reference, String nom, double prix) {  
        this.reference = reference;  
        this.nom = nom;  
        this.prix = prix;  
    }  
  
    // Getters  
    public String getNom() { return nom; }  
    public double getPrix() { return prix; }  
    public String getReference() { return reference; }  
  
    // Essentiel pour fonctionner comme Clé dans une Map  
    @Override  
    public boolean equals(Object o) {  
        if (this == o) return true;  
        if (o == null || getClass() != o.getClass()) return false;  
        Produit produit = (Produit) o;  
        // On se base sur la référence unique  
        return Objects.equals(reference, produit.reference);  
    }  
  
    @Override  
    public int hashCode() {  
        return Objects.hash(reference);  
    }  
}
```

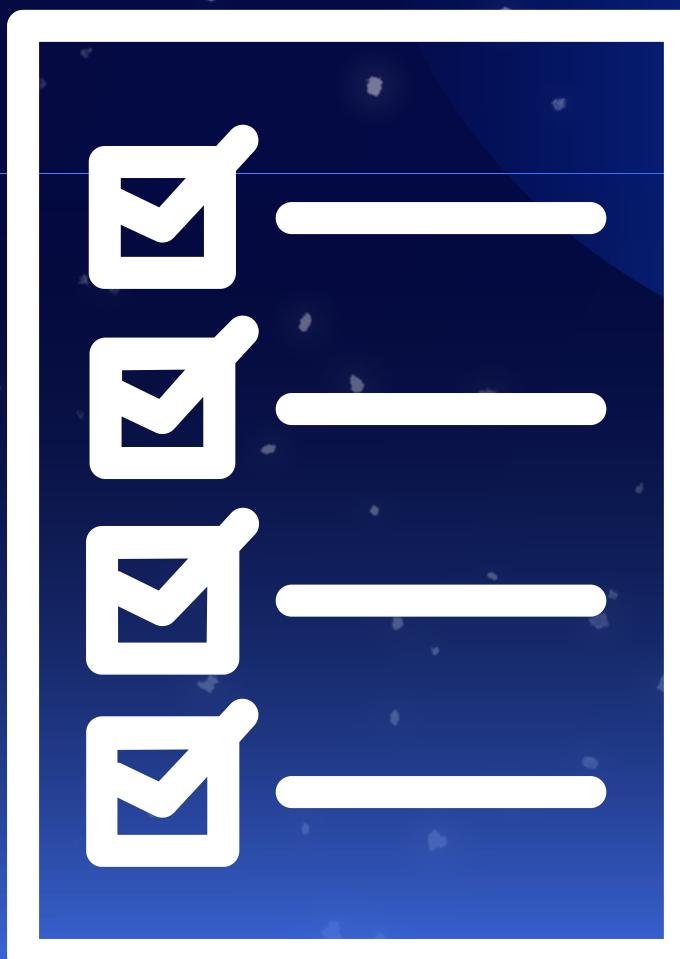
```
public class Panier {  
  
    // L'état est privé et géré DANS la classe  
    private final Map<Produit, Integer> items = new HashMap<>();  
  
    public void ajouterProduit(Produit produit, int quantite) {  
        if (produit == null || quantite <= 0) {  
            return;  
        }  
        this.items.merge(produit, quantite, Integer::sum);  
    }  
  
    // Calcule le prix total du panier.  
    public double getTotalPrix() {  
        double total = 0.0;  
        for (Map.Entry<Produit, Integer> entry : items.entrySet()) {  
            total += entry.getKey().getPrix() * entry.getValue();  
        }  
        return total;  
    }  
  
    public Map<Produit, Integer> getItems() {  
        return Collections.unmodifiableMap(items);  
    }  
  
    public void viderPanier() {  
        items.clear();  
    }  
}
```

L'IA-Driven TDD

Test-Driven Development

division en Deux Tâches :

- 1 Agent Testeur
- 2 Agent Codeur



Démonstration : “Vibe” vs “Piloté”

Implémenter une validation de panier fiable

En mode
“Vibe”

En mode
“Piloté”



Premiere demonstration “Vibe”

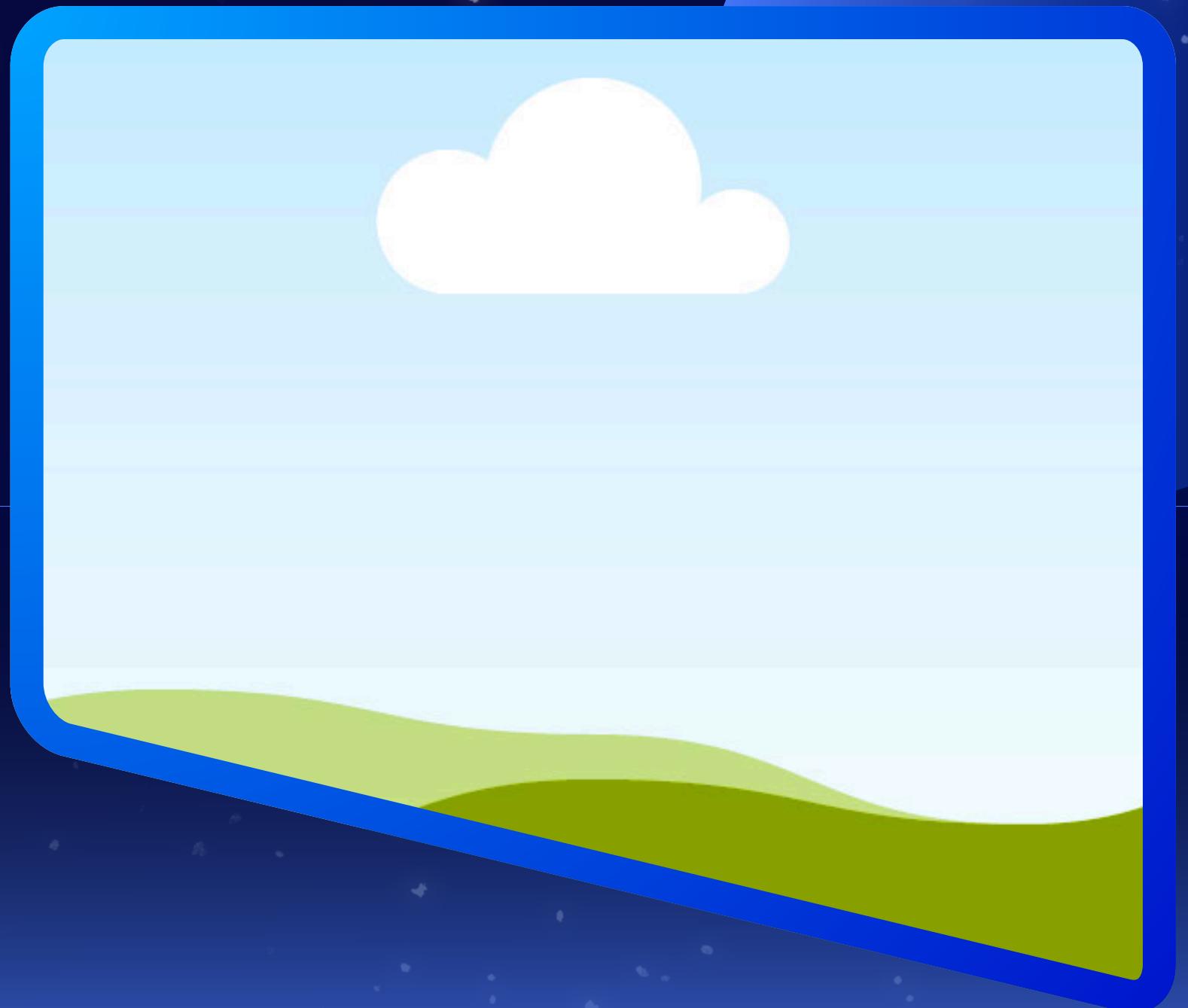
Approche classique
similaire au prompt
ChatGPT

Deuxièmes démonstration

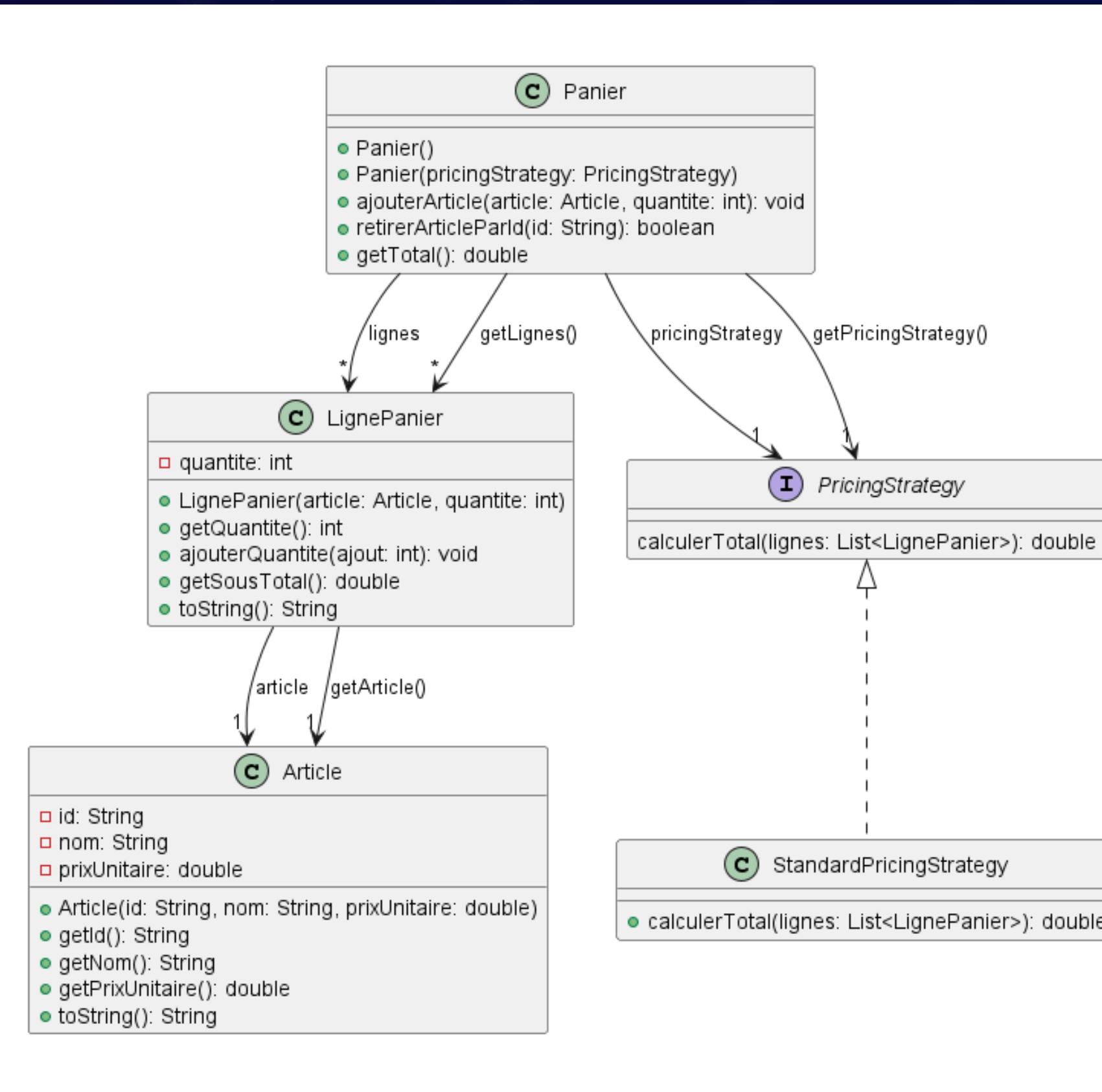
“Piloté”

Utilisation d'un script.
CrewAI pré préparé

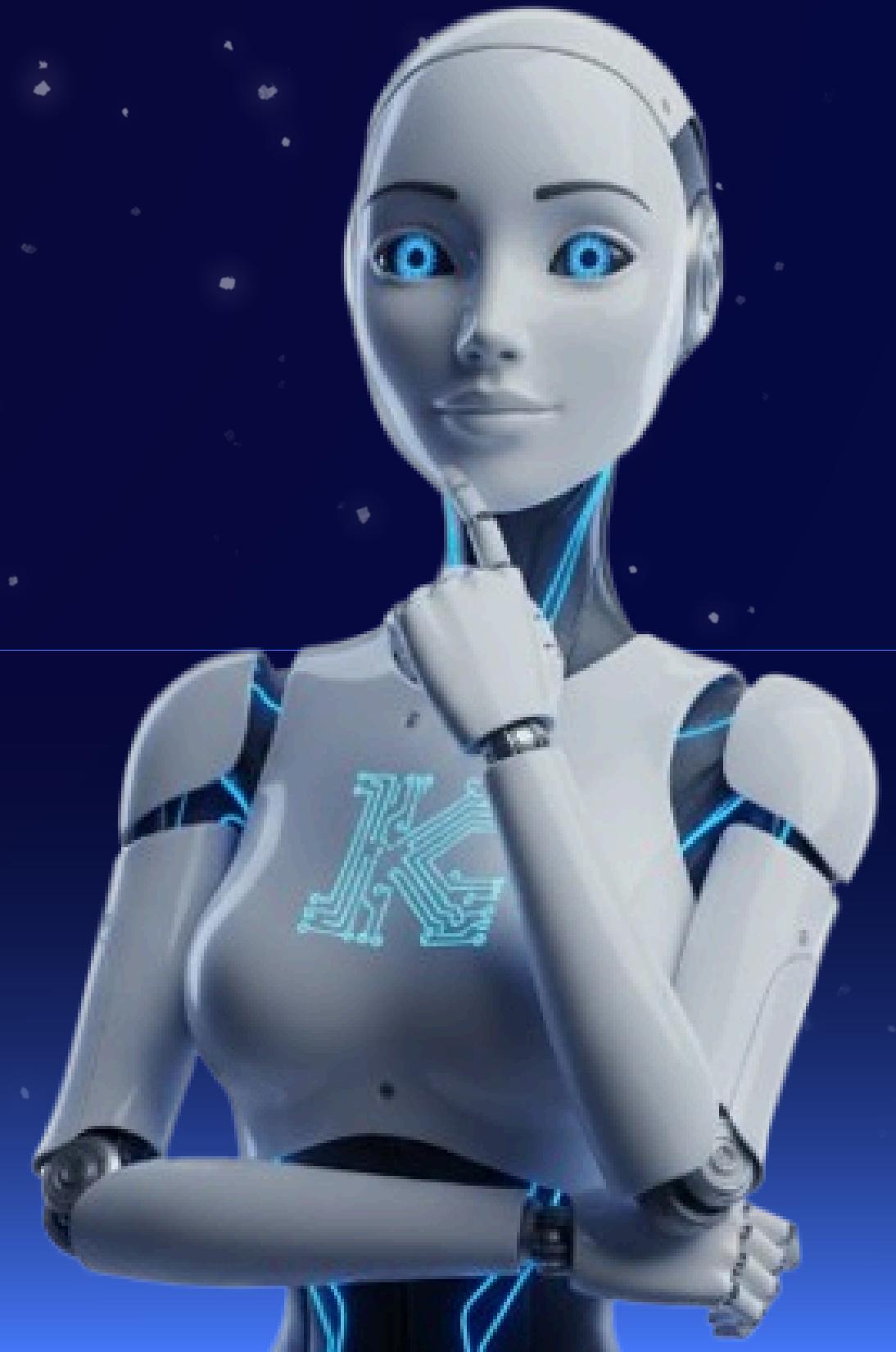
LIVE CODING



Rétroconception du code



La Conclusion Logique



Conclusion

L'IA ne nous
remplace pas,
elle nous
transforme.



Bibliographie

Sources :

<https://www.hexotech.fr/>

<https://www.crewai.com/>

<https://www.adimeo.com/>

<https://docs.github.com/fr/copilot>

Merci de nous
avoir écouté