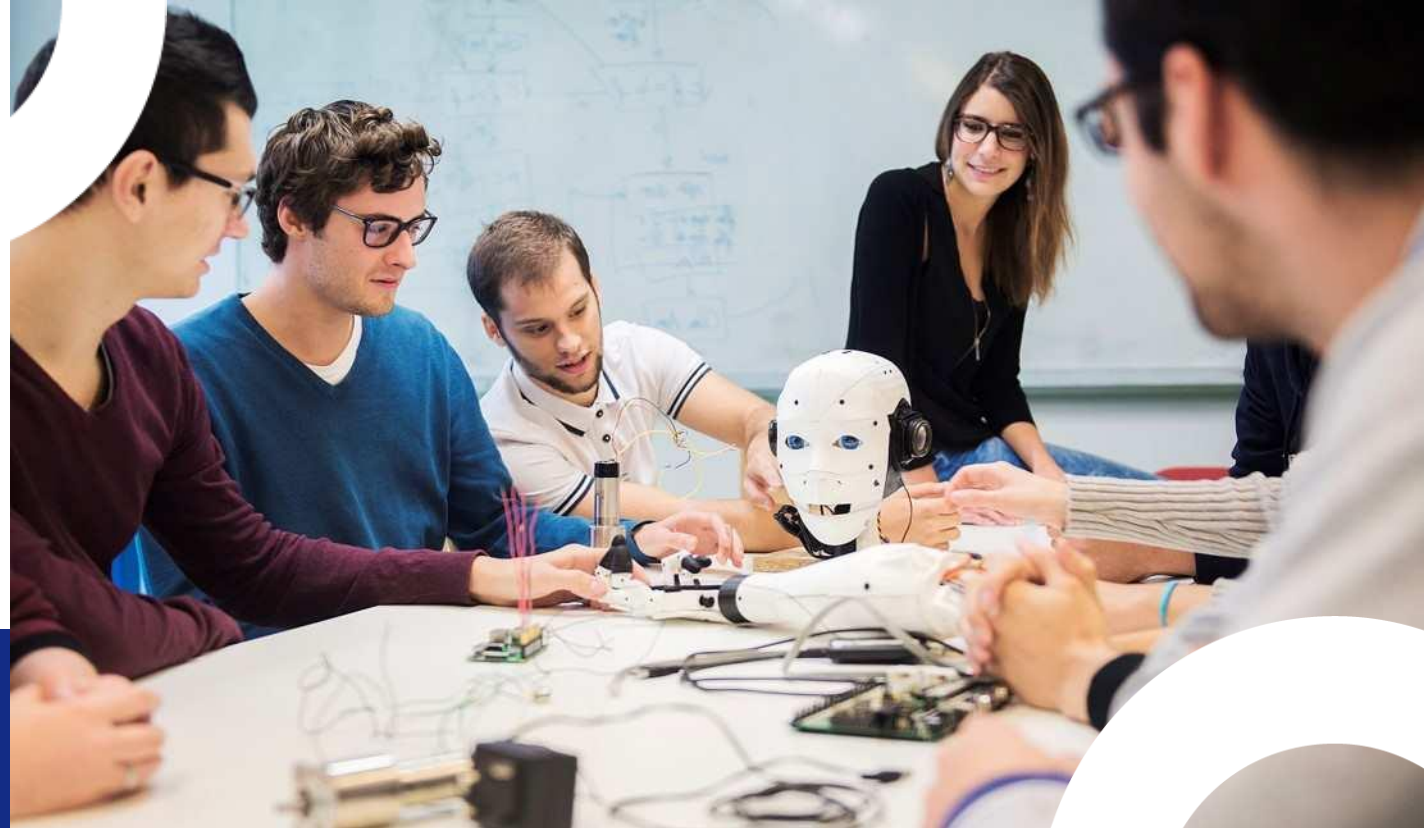




L'ÉCOLE DE L'INNOVATION  
TECHNOLOGIQUE



**MBDA**  
MISSILE SYSTEMS

# Exercice Scientifique et technique Storybook

## ANDRIATSAHAVOJAONA Ony Brunella

Une école de



# Introduction

---

- **Mon contexte chez MBDA**
  - Interfaces web en Angular
  - Objectif : migration de composants JavaScript → Angular
  - Raison : performances, compatibilité, standardisation
- **Pourquoi tester avant d'intégrer ?**
  - Composants réutilisés dans plusieurs IHM
  - Les bugs peuvent se propager
- **Besoin**
  - Tester les composants isolément
  - Vérifier rendu + comportement

**Solution : Storybook**

# Problématique et plan

---

**Problématique :** *En quoi Storybook constitue-t-il un outil pertinent pour le développement, le test et la documentation de composants UI Angular ?*

## Plan :

1. Qu'est-ce que Storybook ? (*Présentation, historique, écosystème*)
2. Architecture et fonctionnement
3. Intégration avec Angular
4. Usages concrets
5. Comparaisons avec d'autres outils
6. Liens avec les cours

# Qu'est-ce que Storybook ?

---

- Outil **open-source** pour développer des composants UI
- Permet de tester un composant **isolément**
- Compatible avec **Angular, React, Vue...**
- Création d'un **catalogue visuel interactif**

## Fonctionnalités principales :

- Développement isolé
- Tests visuels
- Documentation automatique
- Extensions (addons)

# Architecture & concepts clés

## 3 éléments principaux :

- **Stories (.stories.ts)**
  - Scénarios d'affichage d'un composant
  - Différents états
- **Storybook Core**
  - Moteur qui charge et affiche les stories
- **Addons**
  - Extensions : Controls, Actions, Accessibilité, Docs

```
// Button.stories.ts
import { Meta, moduleMetadata, Story } from "@storybook/angular";
import { CommonModule } from "@angular/common";
import { Button } from "../button.component";

export default {
  component: Button,
  decorators: [
    moduleMetadata({
      declarations: [Button],
      imports: [CommonModule],
    })
  ],
} as Meta;

export const Primary: Story = (args) => ({
  props: args,
  template: `<app-button></app-button>`,
});

Primary.args = {
  label: "Button",
  primary: true,
};
```

# Intégration dans un projet Angular

1. **Installation** - configure automatiquement Storybook dans le projet Angular  
`ng add @storybook/angular`
2. **Structure des fichiers** - un fichier de stories par composant, colocalisé avec le code source  
`component.stories.ts`
3. Lancement sur un **serveur local** - interface visuelle immédiate  
`npm run storybook`
4. **Génération pour CI/CD**  
`npm run build-storybook`

# Mes usages concrets chez MBDA

1. Création de **stories pour chaque composant**
2. Tests visuels **indépendants de l'application**
  - gain de temps et détection rapide des anomalies visuelles.
3. Utilisation des addons :
  - **Controls** → modification dynamique des paramètres depuis l'UI
  - **Actions** → suivi des événements (clics, changements) pour valider le comportement
4. **Documentation automatique** des composants
  - génère une page de documentation par composant, avec les props disponibles, leurs types et valeurs par défaut.

# Comparaison avec d'autres outils

Critère	Jest / Karma	Cypress	Doc. manuelle	Storybook
Tests unitaires	✓ Excellent	○ Partiel	✗ Non	○ Partiel
Tests visuels	✗ Non	✓ Oui	✗ Non	✓ Excellent
Isolation composant	○ Partiel	✗ Non	✗ Non	✓ Excellent
Documentation	✗ Non	✗ Non	✓ Manuelle	✓ Auto-générée
Interactivité	✗ Non	✓ Oui	✗ Non	✓ Excellent
Facilité d'usage	○ Moyen	○ Moyen	✓ Simple	✓ Simple

**Conclusion** : Storybook ne remplace pas Jest (tests logiques) ni Cypress (tests E2E), mais comble un manque précis : tester et documenter les composants visuellement de façon isolée.



# Lien avec les cours

---

- **Tests unitaires en Java**
  - Utilisation de JUnit
  - Test d'une méthode isolée
- **Avec Storybook**
  - Isolation d'un composant UI
  - Vérification du rendu et du comportement
- **Même logique appliquée à l'interface**
- **Documentation continue et méthode agile**
  - Principe de "documentation vivante" → toujours synchronisée avec le code, sans effort supplémentaire

# Conclusion

---

- **Storybook facilite :**
  - le développement
  - les tests visuels
  - la documentation
- **Amélioration de la qualité des composants Angular**
- **Mise en pratique de concepts du génie logiciel :**
  - isolation
  - vérification et validation
  - documentation continue
  - réutilisabilité