

# Développer avec GitHub Copilot (Mode Agent)

**Tu expliques → Copilot planifie → Copilot code**

# **Agents Copilot vs chat classique**

## **Agents Copilot : la révolution**

- Les agents = autonomie + planification.
- Mode Plan = décomposition automatique des tâches.
- Exécution multi-étapes sans intervention humaine.
- Mieux que le « prompt » au hasard.

# Développer avec GitHub Copilot (Mode Agent)

Copilot ne code pas “au hasard” — il suit des instructions.

# Introduction : Les modes de GitHub Copilot (vue d'ensemble)

- Ask — questions et explications
- Edit — modifications ciblées
- Agent — réalisation bout en bout
- Plan — planification avant code

# Mode Ask

Sert à poser des questions à Copilot.

Il explique du code, des concepts ou des erreurs.

👉 Exemple : « À quoi sert cette fonction ? » ou « Explique-moi cette erreur »

📌 Il ne modifie pas le code, il répond seulement.

# Mode Edit

Sert à modifier du code existant.

On lui dit quoi changer, et Copilot applique les modifications.

👉 Exemple : « Simplifie cette fonction » ou « Ajoute des commentaires »

📌 Pratique pour corriger, améliorer ou refactoriser du code.

# Mode Agent

Copilot agit comme un développeur autonome.  
Il peut analyser tout le projet, créer ou modifier plusieurs fichiers, et résoudre un problème de bout en bout.

👉 Exemple : « Ajoute une authentification à l'application »

📌 Idéal pour des tâches complexes et globales.

# Mode Plan

Sert à préparer une solution avant de coder.

Copilot propose un plan étape par étape.

👉 Exemple : « Comment ajouter une API REST à ce projet ? »

📌 Utile pour réfléchir à l'architecture et valider l'approche.

# Idée clé à retenir

“

Tu expliques ce que tu veux.  
Copilot réfléchit à comment le faire.  
Copilot écrit le code.

”

# Pourquoi utiliser Copilot comme un agent ?

- Ne pas coder sans vision ni plan
  - Garder une vision claire du besoin
  - Accélérer le développement sans perdre le contrôle
- Copilot devient un assistant de développement, pas juste un auto-compléteur.

# Workflow en 3 étapes

- 1 Décrire le besoin (toi) — une spécification simple, en langage humain**
- 2 Laisser Copilot réfléchir (Mode Plan) — plan technique + découpage**
- 3 Laisser Copilot coder (Agent) — implémentation selon le plan**

# Étape 1 — Écrire de bonnes instructions

Fichier : `.github/copilot-instructions.md`

**Question centrale** : Qu'est-ce que je veux construire, et pourquoi ?

# Contenu recommandé

Dans `.github/copilot-instructions.md` :

- Objectif : à quoi sert l'application ?
- Comportements clés : que peut faire l'utilisateur ?
- Contraintes : règles techniques importantes

# Exemple simple

```
# Application Todo
```

```
## Objectif
```

Créer une application web simple pour gérer des tâches.

```
## Comportements clés
```

- Ajouter une tâche
- Marquer une tâche comme complétée
- Supprimer une tâche
- Sauvegarde locale des données

```
## Contraintes
```

- React + TypeScript
- Tailwind CSS
- Pas de dépendances externes



Pas besoin d'écrire du code, ni un plan technique.

# Étape 2 — Mode Plan (Copilot)

Prompt recommandé : « Analyse ces instructions et propose un plan. »

Copilot va :

- choisir une structure de projet
- définir les composants
- proposer un ordre logique de travail

👉 Ce plan sert à vérifier que vous êtes alignés.

# Étape 3 — Exécution (Copilot Agent)

Une fois le plan validé :

- Copilot écrit le code, étape par étape
- Tu peux interrompre ou corriger à tout moment  
Ton rôle : valider, ajuster, garder la cohérence.

# Rôles

- Toi : définir le besoin, valider le plan, relire le code
- Copilot : transformer l'idée en plan, découper le travail, écrire le code

# Règle d'or

“ Plus l'intention est claire, meilleur sera le code. ”

# Prompts clés

# Pour valider la spec

« Valide cette spec pour clarté et complétude »

# Pour générer le plan

« En Mode Plan, génère un plan technique détaillé »

# Pour découper en tâches

« Découpe le projet en tâches ordonnées »

# Pour implémenter

« Implémente la tâche 1 selon le plan »

# Instructions personnalisées

## avancées

### ## Normes de codage

- ESLint + Prettier obligatoires
- Tests unitaires pour chaque fonction
- TypeScript strict: true

### ## Checklist de revue de code

- Performance : pas de re-render inutiles
- Sécurité : validation des inputs
- Accessibilité : WCAG 2.1 niveau AA

# **SDD + Agents = Avantage décisif**

- Jusqu'à +50 % de vitesse (agent autonome)
- Zéro code « au feeling »
- Pivot rapide : modifiez `.github/copilot-instructions.md`
- Une architecture cohérente à la fin

# Pièges à éviter

- Ne pas trop détailler les instructions (trop verbeux = lent)
- Oublier de valider le plan avant le code
- Négliger les instructions personnalisées

# Timing pour un hackathon

**Heures 0–2** : instructions + plan

**Heures 2–20** : code avec l'agent Copilot

**Heures 20–48** : tests + finition

# Stack et personnalisation

- Agents Copilot : Mode Plan
- VS Code : IDE principal avec personnalisations
- Git : `.github/copilot-instructions.md` dans le repo
- awesome-copilot : templates d'agents

# Ressources et exemples

- Personnalisation de Copilot dans VS Code :  
<https://code.visualstudio.com/docs/copilot/customization/overview>
- awesome-copilot : [github.com/github/awesome-copilot](https://github.com/github/awesome-copilot)

# Bonus : structure recommandée

## .github/copilot-instructions.md

```
# [Nom du projet] - Spec & Instructions
```

### ## 1. Objectif

[Description courte : 2-3 phrases]

### ## 2. Motivations

- [Pourquoi ce projet]
- [Cas d'usage principal]

### ## 3. Comportements clés

- [Comportement 1]
- [Comportement 2]
- [Comportement 3]

# Bonus : structure recommandée (suite)

```
## 4. Tech Stack
- **Frontend** : [Framework + version]
- **Backend** : [Framework + version]
- **Database** : [Type + version]
## 5. Architecture
src/
└── components/
└── pages/
└── services/
└── utils/
## 6. Contraintes
- Performance : [X ms]
- Taille du bundle : [X kb]
```

# Démo — Finance dashboard

- Méthode : Spécification → Plan → Code (mode Agent).
- Copilot : structuration du plan et assistance au développement.
- Moi : définition du besoin, validation, tests et corrections.
- Démo : dashboard Plotly, UI chatbot, API FastAPI avec données mock.

# Questions ?

Merci ! 🙏



**Slides** : Lien GitHub Pages



**Repo** : Lien GitHub

# Liens utiles

- Demande d'accès à GitHub Copilot chez Cegid :  
[https://devsecopscegid.atlassian.net/servicedesk/customer/portal/1/user/login?  
destination=portal%2F1%2Fgroup%2F4%2Fcreate%2F43](https://devsecopscegid.atlassian.net/servicedesk/customer/portal/1/user/login?destination=portal%2F1%2Fgroup%2F4%2Fcreate%2F43)
- Cegid Design System : <https://cds-website.azurewebsites.net/guidelines/installation>