SDD + Agents Copilot

Spécification → Plan → Code (autonome)

Mode Plan pour les hackathons

Pragmatique • Autonome • Rapide

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.

SDD + Agents Copilot

La spec définit → l'agent exécute

- Spécifications structurées = un brief pour l'agent.
- L'agent lit, comprend et planifie.
- Mode Plan = stratégie avant le code.

Workflow des agents : 3 étapes

1. **Spécification** — Rédiger le brief
2. **Mode Plan** — L'agent décompose
3. **Exécution** — L'agent code

Étape 1 : Spécification

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

- Idée : en 2–3 phrases
- Motivations : pourquoi c’est important
- Comportements clés : 3–5 cas d’usage

Exemple d’instructions

`.github/copilot-instructions.md`

```
# Spécification de l’application Todo

## Objectif
Une app simple pour gérer des tâches.
Ajouter, cocher, supprimer des items.
## Comportements clés
- L’utilisateur crée une tâche
- Marque comme complétée
- Supprime des tâches
- Persistance via localStorage
## Stack requis
- React + TypeScript
- Tailwind CSS
- Pas de dépendances externes
```

Étape 2 : Plan technique

Fichier : `.github/copilot-instructions.md` (section PLAN)

- Stack technique : framework, base de données, etc.
- Architecture : structure des fichiers
- Contraintes : performance, sécurité

Étape 3 : Tâches

Fichier : `.github/copilot-instructions.md` (section TASKS)

- Diviser le projet en 3 à 8 tâches
- Chaque tâche : 30 à 60 min
- Ordre logique selon les dépendances

Utiliser les agents Copilot

- Ouvrir VS Code + Copilot Chat
- Sélectionner Mode Plan
- L'agent lit automatiquement `.github/copilot-instructions.md`
- L'agent génère un plan puis code de manière autonome

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`
- Le jury voit une architecture cohérente

Pièges à éviter

- Ne pas trop détailler les instructions (trop verbeux = lent)
- Oublier de valider le plan avant le code
- Changer la spec sans mettre à jour les tâches
- Négliger les instructions personnalisées

Timing pour un hackathon de 48 h

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
- Instructions personnalisées : https://code.visualstudio.com/docs/copilot/customization/custom-instructions
- awesome-copilot : github.com/github/awesome-copilot
- Compétences des agents : capacités extensibles

Construisons ! 🚀

SDD + Agents Copilot = produit fini en temps record

```
git commit .github/copilot-instructions.md
git checkout -b feature/copilot-sdd
```

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]
```

Bonus : tâches et normes

```
## 7. Tâches (ordre logique)
1. Setup & structure
2. Core models
3. API/backend
4. UI components
5. Integration
6. Tests
7. Polish & optimisations

## 8. Normes de codage
- [Linter + config]
- [Type checking : TypeScript strict]
- [Testing : Unit + E2E]
```

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
- Cegid Design System : https://cds-website.azurewebsites.net/guidelines/installation