

SDD + Copilot Agents

Spec → Plan → Code (Autonome)

Mode Plan pour Hackathons

Pragmatique • Autonome • Rapide

Copilot Agents vs Chat classique

Copilot Agents: La Révolution

- Les agents = autonomie + planification
- Mode Plan = décomposition automatique
- Exécution multi-étapes sans intervention
- Meilleur que "prompt aléatoire"

SDD + Copilot Agents

Spec définit → Agent exécute

- Spécifications structurées = brief pour l'agent
- Agent lit + comprend + planifie
- Mode Plan = stratégie avant code

Workflow Agents: 3 Étapes

- 1. Spec** - Écrivez le brief
- 2. Mode Plan** - Agent décompose
- 3. Execute** - 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 Instructions

.github/copilot-instructions.md

Todo App Spec

Objectif

Une app simple pour gérer des tâches.
Ajouter, cocher, supprimer des items.

Comportements clés

- L'utilisateur crée une tâche
- Marque comme complétée
- Supprime les tâches
- Persiste en localStorage

Stack requis

- React + TypeScript
- Tailwind CSS

Étape 2: Plan Technique

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

- **Tech Stack:** Framework, DB, etc
- **Architecture:** Structure fichiers
- **Contraintes:** Performance, sécurité

Étape 3: Tâches

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

- Divisez le projet en 3-8 tâches
- Chaque tâche: 30-60 min
- Ordre logique de dépendances

Utiliser Copilot Agents

- Ouvrez VSCode + Copilot Chat
- Sélectionnez **Mode Plan** (astérisque *)
- L'agent lit `.github/copilot-instructions.md` automatiquement
- L'agent génère plan → code 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"

Custom Instructions Avancées

Coding Standards

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

Code Review Checklist

- Performance: pas de re-renders inutiles
- Sécurité: validation des inputs
- Accessibilité: WCAG 2.1 Level AA

SDD + Agents = Cheat Code

- +50% vitesse (agent autonome)
- Zéro "vibe coding"
- Pivot rapide: modifiez `.github/copilot-instructions.md`
- Jury voit une archi cohérente

Pièges à Éviter

- Ne pas trop détailler les instructions (verbex = lent)
- Oublier de valider le plan avant code
- Changer spec sans maj des tâches
- Négliger les custom instructions

Timing pour Hackathon 48h

Heure 0-2: Instructions + Plan

Heure 2-20: Copilot Agent code

Heure 20-48: Tests + Polish

Stack + Customization

- **Copilot Agents**: Mode Plan (*)
- **VSCODE**: IDE principal avec customizations
- **Git**: `.github/copilot-instructions.md` en repo
- **awesome-copilot**: Templates agents

Ressources + Exemples

- **VSCode Copilot Customization:**

<https://code.visualstudio.com/docs/copilot/customization/overview>

- **Custom Instructions:**

<https://code.visualstudio.com/docs/copilot/customization/custom-instructions>

- **awesome-copilot:** github.com/github/awesome-copilot

- **Agent Skills:** Extensible capabilities

Let's Build!

SDD + Copilot Agents = 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]
- Bundle size: [X kb]
```

Bonus: Tasks & Standards

7. Tasks (Ordre logique)

1. Setup & structure
2. Core models
3. API/backend
4. UI components
5. Integration
6. Tests
7. Polish & optimizations

8. Coding Standards

- [Linter + config]
- [Type checking: TypeScript strict]
- [Testing: Unit + E2E]

Questions ?

Merci ! 🙏



Slides: [Lien GitHub Pages]



Repo: [Lien GitHub]