le workflow 3 agents

je voudrais que tu reflechisses en tant que leader chief architect. tu veux
repartir 3 claude agent sur trois terminaux et leur permettre de travailler sur
cette todo list en paralleles.
est ce que tu vas diviser , en tenant compte des frictions eventuelles, la
todolist en differentes actions complementaires , reparties sur les trois
claude-terminaux , pour qu ils puissent travailler sous mon controle, mais de
facon simultanée et parallele.
ou tu penses que ce n'est pas souhaitable, a cause des frictions possibles en
travaillant sur les memes fichiers.
ou tu as une meilleure idée?
prends ton temps. reflechis bien respire bien. deep think

* Thinking…

L'utilisateur me demande de réfléchir en tant que lead architect pour organiser
le travail de 3 agents Claude en parallèle sur la todolist du système de thèmes.
Je dois analyser :

1. Les dépendances entre les tâches
2. Les risques de conflits de fichiers
3. Les opportunités de parallélisation
4. Une stratégie optimale de répartition

Analysons la todolist :

- theme-1 : themeStore.ts (nouveau fichier)
- theme-2 : détection système macOS (Tauri backend)
- theme-3 : themes.css (nouveau fichier)
- theme-4 : ThemeToggle.tsx (nouveau fichier)
- theme-5 : classes dans index.css (fichier existant)
- theme-6 : test sur App.tsx (fichier existant)
- theme-7 : glassmorphism adaptation (index.css)

- theme-8 : intégration preferencesStore (fichiers existants)
- theme-9 : SettingsPanel (fichier existant)
- theme-10 : adaptation composants (multiples fichiers)
- theme-11 : syntaxe highlighting (fichiers CSS)
- theme-12 : persistance (test)

Zones de friction potentielles :

- index.css : touché par theme-5 et theme-7
- App.tsx : touché par theme-6
- preferencesStore : touché par theme-8
- SettingsPanel : touché par theme-9

Opportunités de parallélisation :

- Nouveaux fichiers peuvent être créés en parallèle sans conflit
- Backend Rust et frontend peuvent avancer indépendamment
- CSS et composants React sont découplés

Je vais proposer une répartition intelligente qui minimise les frictions.

> En tant que Lead Architect, voici mon analyse approfondie :
>
> Analyse des dépendances et frictions

Zones de friction identifiées :

1. index.css - Touché par plusieurs tâches (theme-5, theme-7)
2. App.tsx - Point d'intégration central (theme-6)
3. preferencesStore.ts - Système existant critique (theme-8)
4. Backend Rust - Nécessite compilation (theme-2)

Opportunités de parallélisation :

- Création de nouveaux fichiers isolés (aucun conflit)
- Séparation frontend/backend
- Modules CSS indépendants

> Stratégie recommandée : 3 tracks parallèles

Terminal 1 : Frontend Foundation (Agent Alice)

Responsabilité : Créer l'infrastructure de base du thème

- theme-1 : Créer themeStore.ts ✓
- theme-3 : Créer themes.css avec variables ✓

- theme-5 : Ajouter classes .theme-light dans index.css ✓
  Avantage : Travaille principalement sur de nouveaux fichiers

Terminal 2 : Backend & System (Agent Bob)

Responsabilité : Intégration système et persistance

- theme-2 : Détection thème macOS (Rust/Tauri) ✓
- theme-8 : Préparer structure préférences (analyse seulement)
- theme-12 : Planifier tests de persistance
  Avantage : Isolé du frontend, peut compiler indépendamment

Terminal 3 : UI Components (Agent Charlie)

Responsabilité : Composants visuels

- theme-4 : Créer ThemeToggle.tsx ✓
- theme-7 : Documenter adaptations glassmorphism nécessaires
- theme-11 : Préparer palette syntaxe highlighting
  Avantage : Nouveaux composants, préparation visuelle

  Workflow de synchronisation

Phase 1 (Parallèle - 30 min)

- Chaque agent travaille sur ses tâches isolées
- Aucun conflit possible

Point de sync 1

- Review des créations
- Merge du themeStore + themes.css + ThemeToggle

Phase 2 (Semi-parallèle - 20 min)

- Agent Alice : Intègre theme-6 (test sur App.tsx)
- Agent Bob : Implémente theme-8 (préférences)
- Agent Charlie : Continue documentation visuelle

Point de sync 2

- Validation de l'intégration
- Résolution des conflits éventuels

  Alternative recommandée : Pipeline séquentiel optimisé

Si les frictions vous inquiètent, je recommande plutôt :

Un seul agent principal + 2 agents en support :

- Agent principal : Suit la todolist dans l'ordre
- Agent support 1 : Prépare les assets (icônes, palettes couleurs)
- Agent support 2 : Écrit les tests et documentation

Cette approche élimine 95% des frictions tout en gardant de la parallélisation.

Ma recommandation finale

Option Pipeline séquentiel - Plus sûre, moins de coordination nécessaire,
progression visible. Les agents support préparent le terrain sans toucher au
code.

Qu'est-ce qui vous semble le plus adapté à votre style de travail ?