Planning Projet Somnium - 4 jours (Détaillé)

JOUR 1 - Fondations du Système

🟢 Développeur 1 - Architecture Core

Fichiers assignés : `main.go`, `character/character.go`, `character/creation.go`, `game/menu.go`

⏰ Matin (3h)

1. Setup Projet (30min)

```bash
cd src/
go mod init somnium
```

Structure des dossiers selon consignes

2. `character/character.go` (90min)

```go
```

```go
// Structs à créer
type Character struct {
    Name       string
    Class      string
    Level      int
    MaxHP      int
    CurrentHP  int
    Inventory  []string
    Money      int
    Skills     []string
    Mana       int
    MaxMana    int
    Equipment  Equipment
}

type Equipment struct {
    Head  string
    Chest string
    Feet  string
}

// Fonctions obligatoires à implémenter
func InitCharacter(name, class string) Character
func (c *Character) DisplayInfo()
func (c *Character) IsDead() bool
func (c *Character) Resurrect()
```

3. `main.go` (60min)

```go
// Structure du main
func main() {
    // Appel menu principal seulement
    game.MainMenu()
}
```

⏰ Après-midi (4h)

4. `character/creation.go` (120min)

```go
```

```
// Fonction principale
func CharacterCreation() Character

// Fonctions utilitaires à créer
func validateName(input string) string
func normalizeString(s string) string
func selectClass() string
func getBaseStats(class string) (maxHP, mana int)
```

Logique attendue :

- Validation nom (lettres uniquement)
- Format : Première majuscule, reste minuscule
- Classes : Humain (100HP, 50Mana), Elfe (80HP, 80Mana), Nain (120HP, 30Mana)
- CurrentHP = 50% MaxHP au départ
- Skills de base : ["Coup de poing"]

5. `game/menu.go` (120min)

```
go

// Menu principal avec switch-case
func MainMenu()
func displayMenuOptions()
func handleUserInput() int

// Options du menu
// 1. Afficher informations personnage
// 2. Accéder à l'inventaire
// 3. Marchand
// 4. Forgeron
// 5. Entrainement
// 6. Qui sont-ils
// 7. Quitter
```

🟡 Développeur 2 - Systèmes Combat

Fichiers assignés : `combat/monster.go`, `combat/spells.go`, `character/inventory.go`

⏰ Matin (3h)

1. `combat/monster.go` (90min)

```
go
```

```
// Struct obligatoire
type Monster struct {
    Name      string
    MaxHP     int
    CurrentHP int
    Attack    int
}

// Fonction obligatoire
func InitGoblin() Monster
```

Specs Gobelin :

- Name: "Gobelin d'entrainement"
- MaxHP: 40
- CurrentHP: 40
- Attack: 5

2. `combat/spells.go` (90min)

```
go

// Fonctions de sorts
func CoupDePoing(caster *character.Character, target *Monster) int
func BouleDeFeu(caster *character.Character, target *Monster) int
func (c *Character) CanCastSpell(spellName string) bool
func (c *Character) ConsumeMP(cost int) bool

// SpellBook fonction
func (c *Character) LearnSpell(spellName string) bool
```

Dégâts attendus :

- Coup de poing: 8 dégâts, 5 mana
- Boule de feu: 18 dégâts, 15 mana

⏰ Après-midi (4h)

3. `character/inventory.go` (240min)

```
go
```

```
// Fonctions inventory obligatoires
func (c *Character) AccessInventory()
func (c *Character) TakePot() bool
func (c *Character) TakePoison() bool
func (c *Character) AddToInventory(item string) bool
func (c *Character) RemoveFromInventory(item string) bool
func (c *Character) HasInventorySpace() bool
func (c *Character) CountItem(itemName string) int

// Constantes
const MaxInventorySlots = 10
```

Logique attendue :

- TakePot: +50 HP, consomme 1 potion
- TakePoison: -10 HP/seconde pendant 3 secondes
- Limite 10 items de base

🔵 Développeur 3 - Interface & Utilitaires

Fichiers assignés : `shop/merchant.go`, fonctions utilitaires diverses

⏰ Matin (3h)

1. `shop/merchant.go` (180min)

```
go

// Interface marchand
func MerchantMenu(player *character.Character)
func displayMerchantItems()
func processPurchase(player *character.Character, itemChoice int) bool
func canAfford(player *character.Character, price int) bool

// Items et prix obligatoires
var MerchantItems = map[string]int{
    "Potion de vie":          3,
    "Potion de poison":       6,
    "Livre de Sort: Boule de feu": 25,
    "Fourrure de Loup":       4,
    "Peau de Troll":          7,
    "Cuir de Sanglier":       3,
    "Plume de Corbeau":       1,
    "Augmentation d'inventaire": 30,
}
```

⏰ Après-midi (4h)

2. Fonctions utilitaires dans divers fichiers (240min)

Dans `character/character.go` - Ajouts:

```go
func (c *Character) PoisonEffect()  // 3 secondes, -10HP/sec avec time.Sleep
func (c *Character) DisplayInventory()
func (c *Character) UseItem(itemName string) bool
```

Tests et intégration:

- Tester toutes les fonctions créées
- S'assurer que les imports fonctionnent
- Créer des messages d'erreur clairs

JOUR 2 - Systèmes Économiques & Combat

🟢 Développeur 1 - Économie & Craft

Fichiers assignés : `shop/forge.go`, ajouts `character/character.go`

⏰ Matin (3h)

1. `shop/forge.go` (180min)

```go

```

```go
// Interface forgeron
func ForgeMenu(player *character.Character)
func displayCraftableItems()
func craftItem(player *character.Character, itemChoice int) bool
func hasRequiredMaterials(player *character.Character, recipe Recipe) bool

// Struct recette
type Recipe struct {
    Name      string
    Materials map[string]int  // "materiau": quantité
    Cost      int
    Result    string
}

// Recettes obligatoires
var Recipes = []Recipe{
    {
        Name: "Chapeau de l'aventurier",
        Materials: map[string]int{"Plume de Corbeau": 1, "Cuir de Sanglier": 1},
        Cost: 5,
        Result: "Chapeau de l'aventurier",
    },
    {
        Name: "Tunique de l'aventurier",
        Materials: map[string]int{"Fourrure de Loup": 2, "Peau de Troll": 1},
        Cost: 5,
        Result: "Tunique de l'aventurier",
    },
    {
        Name: "Bottes de l'aventurier",
        Materials: map[string]int{"Fourrure de Loup": 1, "Cuir de Sanglier": 1},
        Cost: 5,
        Result: "Bottes de l'aventurier",
    },
}
```

⏰ Après-midi (4h)

2. Système d'équipement dans `character/character.go` (240min)

```go
```

```
// Nouvelles méthodes Character
func (c *Character) EquipItem(itemName string) bool
func (c *Character) UnequipItem(slot string) bool
func (c *Character) GetEquipmentBonus() int
func (c *Character) UpdateStatsFromEquipment()
func (c *Character) DisplayEquipment()

// Bonus d'équipement obligatoires
// Chapeau: +10 MaxHP
// Tunique: +25 MaxHP
// Bottes: +15 MaxHP
```

3. Système upgrade inventaire

```
go

func (c *Character) UpgradeInventorySlot() bool
// MaxSlots + 10, max 3 fois, coût 30 or
```

🟡 Développeur 2 - Combat Avancé

Fichiers assignés : `combat/fight.go`, améliorations combat

⏰ Matin (3h)

1. `combat/fight.go` (180min)

```
go

// Combat principal
func TrainingFight(player *character.Character)
func CharacterTurn(player *character.Character, monster *Monster, turn int) bool
func GoblinPattern(goblin *Monster, player *character.Character, turn int)

// Variables de combat
type CombatState struct {
    Turn        int
    PlayerAlive   bool
    MonsterAlive  bool
}

// Logique GoblinPattern
// Tour normal: 100% attack (5 dégâts)
// Tous les 3 tours: 200% attack (10 dégâts)
```

⏰ Après-midi (4h)

2. Menu combat dans CharacterTurn (120min)

```go
// Options combat
// 1. Attaquer (attaque basique 5 dégâts)
// 2. Sorts (si mana suffisant)
// 3. Inventaire (potions)
// 4. Fuir (retour menu)
```

3. MISSION 1 - Initiative (120min)

```go
// Ajouter Initiative aux structs
// Character.Initiative et Monster.Initiative
func DetermineFirstPlayer(player *character.Character, monster *Monster) bool
func RollInitiative() int  // Random 1-20
```

🔵 Développeur 3 - Polish & UX

Fichiers assignés : Amélioration UX, messages, validation

⏰ Matin (3h)

1. Messages d'erreur personnalisés (180min)

```go
// Dans chaque fichier, créer des constantes de messages
const (
    ErrNotEnoughMoney    = "💰 Pas assez d'argent! Il vous faut %d pièces d'or."
    ErrInventoryFull     = "🎒 Inventaire plein! Vendez ou utilisez des objets."
    ErrNotEnoughMaterials = "⚒️ Matériaux insuffisants pour %s"
    ErrNotEnoughMana     = "🔮 Pas assez de mana! (%d requis)"
    ErrItemNotFound      = "❌ Objet '%s' introuvable dans l'inventaire"
)

// Fonction d'affichage coloré (optionnel)
func PrintError(message string)
func PrintSuccess(message string)
func PrintInfo(message string)
```

⏰ Après-midi (4h)

2. Validation inputs et navigation (240min)

```go
// Fonctions utilitaires globales
func GetUserInput(prompt string) string
func GetUserChoice(min, max int) int
func ValidateInput(input string, validOptions []string) bool
func ClearScreen()  // Optionnel pour meilleure UX
func PressEnterToContinue()

// Améliorer tous les menus avec ces validations
```

JOUR 3 - Missions Bonus & Finition

🟢 Développeur 1 - Missions Avancées

Fichiers assignés : `game/dungeon.go`, MISSION 2 (XP)

⏰ Matin (3h)

1. MISSION 2 - Système d'expérience (180min)

```go
// Ajouter à Character struct
type Character struct {
    // ... existant
    CurrentXP int
    MaxXP     int
    XPUpgrades int  // compteur montées niveau
}

// Fonctions XP
func (c *Character) GainXP(amount int)
func (c *Character) CheckLevelUp() bool
func (c *Character) LevelUp()
func (c *Character) CalculateXPNeeded(level int) int

// XP par monstre
// Gobelin: 25 XP
// XP nécessaire: Level * 100
// Bonus level up: +20 MaxHP, +10 MaxMana, +5 toutes stats combat
```

⏰ Après-midi (4h)

2. `game/dungeon.go` - MISSION 5 (240min)

```go
// Système de donjons créatif
func DungeonExploration(player *character.Character)
func GenerateRandomEvent() Event
func ProcessEvent(player *character.Character, event Event)

type Event struct {
    Name        string
    Description string
    Type        string  // "combat", "treasure", "trap", "merchant"
    Effect      func(*character.Character)
}

// Événements créatifs liés à l'univers Somnium
// - Rêves lucides (bonus temporaires)
// - Cauchemars (malus)
// - Fragments de mémoire (histoire)
// - Marchands oniriques
```

🟡 Développeur 2 - Combat Final

Fichiers assignés : MISSION 3 et 4 (Combat magique + Mana)

⏰ Matin (3h)

1. MISSION 4 - Système Mana complet (180min)

```go
// Dans combat/spells.go - Compléter
func (c *Character) RestoreMana(amount int)
func (c *Character) ManaCost(spellName string) int

// Coûts mana obligatoires
var SpellCosts = map[string]int{
    "Coup de poing": 5,
    "Boule de feu":  15,
    "Soin":          10,  // MISSION 3
    "Bouclier":      8,   // MISSION 3
}

// Ajouter Potion de Mana au marchand (8 pièces d'or, +30 mana)
```

⏰ Après-midi (4h)

2. MISSION 3 - Combat magique étendu (240min)

```go
// Nouveaux sorts à implémenter
func Soin(caster *character.Character) int       // +25 HP au lanceur
func Bouclier(caster *character.Character) bool  // Réduit dégâts de 50% pendant 3 tours
func Empoisonnement(target *Monster) bool        // -5 HP/tour pendant 3 tours

// Améliorer patterns des monstres
func GoblinSpecialAttack(goblin *Monster, player *character.Character)
// Attaques spéciales selon le tour

// Ajouter livres de sorts au marchand
```

🔵 Développeur 3 - Documentation & Tests

Fichiers assignés : `README.md`, MISSION 6, tests

⏰ Matin (3h)

1. MISSION 6 - Easter eggs (60min)

```go
// Dans game/menu.go, option "Qui sont-ils"
func DisplayHiddenArtists()

// Les 2 artistes cachés sont dans les titres des tâches :
// Partie 2: "Two for the Price of One" = Abba
// Partie 2: "Gimme! Gimme! Gimme!" = Abba
// Partie 3: "A.I. Intelligence Artificielle" = Spielberg
```

2. README.md complet (120min)

```markdown
# Structure attendue complète
## Description du concept Somnium
## Installation détaillée
## Guide d'utilisation
## Fonctionnalités implémentées (checklist)
## Missions bonus complétées
## Crédits équipe
```

⏰ Après-midi (4h)

3. Tests complets & debug (240min)

- Test de chaque menu
- Test des cas d'erreur
- Vérification limites (argent, mana, HP)
- Test des crafts complets
- Test du système de combat
- Documentation des bugs trouvés

JOUR 4 - Finalisation & Préparation Oral

🟢 Développeur 1 - Code Review & Optimisation

⏰ Matin (2h)

- Code review complet
- Optimisation performance
- Refactoring si nécessaire
- Préparation arguments techniques pour l'oral

⏰ Après-midi (3h)

- Répétition démonstration technique
- Préparation réponses questions jury
- Finalisation repository

🟡 Développeur 2 - Équilibrage & Demo Combat

⏰ Matin (2h)

- Équilibrage final (HP, mana, dégâts, coûts)
- Tests de combat intensifs
- Préparation scénarios de démonstration combat

⏰ Après-midi (3h)

- Répétition démonstration combat
- Tests des cas limites
- Backup des saves de démonstration

🔵 Développeur 3 - Présentation & Communication

⏰ Matin (2h)

- Finalisation documentation
- Préparation support visuel pour oral
- Tests utilisateur avec personnes externes

⏰ Après-midi (3h)

- Répétition présentation complète
- Préparation pitch (concept Somnium)
- Coordination équipe pour l'oral

📋 Checklist de Validation par Fonction

Fonctions Critiques à Valider

go

// character/character.go
✓ InitCharacter() - Initialisation correcte selon classe
✓ DisplayInfo() - Affichage complet stats
✓ IsDead() / Resurrect() - Gestion mort/résurrection

// character/creation.go
✓ CharacterCreation() - Validation nom + choix classe
✓ normalizeString() - Format nom correct

// character/inventory.go
✓ TakePot() - +50HP, limite MaxHP
✓ TakePoison() - -10HP/sec × 3sec
✓ AddToInventory() - Respect limite 10 items
✓ RemoveFromInventory() - Suppression correcte

// shop/merchant.go
✓ MerchantMenu() - Tous items aux bons prix
✓ processPurchase() - Vérif argent + ajout inventory

// shop/forge.go
✓ craftItem() - Vérif matériaux + argent + craft
✓ hasRequiredMaterials() - Validation recettes exactes

// combat/fight.go
✓ TrainingFight() - Boucle combat complète
✓ GoblinPattern() - Attaques normales et spéciales tour 3
✓ CharacterTurn() - Menu combat fonctionnel

// combat/spells.go
✓ CoupDePoing() - 8 dégâts, 5 mana
✓ BouleDeFeu() - 18 dégâts, 15 mana
✓ ConsumeMP() - Vérification mana suffisant

Tests de Non-Régression

1. **Économie :** Acheter sans argent → erreur
2. **Inventaire :** Ajouter au delà de 10 → erreur
3. **Combat :** HP à 0 → mort → résurrection 50%
4. **Craft :** Craft sans matériaux → erreur
5. **Mana :** Sort sans mana → erreur

🎯 Objectif Final Mesurable

✅ **100% Tâches obligatoires** (1-22) ✅ **3+ Missions bonus** sur 6
✅ **Zéro bugs** sur les fonctions critiques ✅ **Démo fluide** 5-10 minutes ✅ **Code propre** et commenté
✅ **Repository** conforme aux consignes