🚀 Galilée OS

Système de Productivité Étudiant Intelligent

Progressive Web Application complète pour la gestion de projets, révisions IA et gamification

🌐 **Application Live** • 📖 **Documentation** • 💻 **GitHub Repository**

👨‍💻 Auteur

BENHAMMADA Ahmed Amine
Étudiant Ingénieur en Informatique - ING2
Université Sorbonne Paris Nord - Sup Galilée
@KESHRUD

📋 Table des Matières

🎯 À Propos du Projet

Contexte Académique

Galilée OS a été développé dans le cadre du cours de **Programmation Web Avancée (PWA)** du programme d'ingénierie informatique ING2 à Sup Galilée, année universitaire 2024-2025.

Vision

L'objectif est de créer une plateforme unifiée qui répond aux besoins réels des étudiants ingénieurs :

Problème	Solution Galilée OS
🗂️ Gestion de multiples projets	Tableau Kanban intuitif avec drag & drop
🧠 Révisions avant examens	Flashcards générées automatiquement par IA
😴 Perte de motivation	Système de gamification avec XP et rangs
📱 Besoin de mobilité	PWA installable fonctionnant hors-ligne
⏰ Gestion du temps	Timer Pomodoro intégré

Pourquoi "Galilée OS" ?

Le nom rend hommage à **Galilée**, le scientifique, et à **Sup Galilée**, l'école d'ingénieurs. Le terme "OS" (Operating System) reflète l'ambition de créer un système complet pour la productivité étudiante.

🖼️ Démonstration

🌐 Application Live

URL de Production : https://galilee-os.netlify.app

Thèmes Disponibles

Thème Galilée (Sci-Fi)	Thème Professionnel
Design futuriste avec effets néon cyan	Interface épurée et minimaliste
Fond 3D animé avec Three.js	Couleurs sobres pour la concentration
Icônes et badges stylisés	Mode classique Kanban

✨ Fonctionnalités Détaillées

1. 📋 Tableau Kanban Complet

Gestion des Colonnes

```
interface Column {
  id: string;
  title: string;      // Nom personnalisable
  order: number;      // Position dans le board
```

```
  color?: string;     // Couleur de la colonne
}
```

Fonctionnalité	Détail Technique
Création	Modal de création avec validation
Édition	Double-clic pour renommer inline
Suppression	Confirmation + migration des tâches
Réorganisation	Drag & drop entre colonnes
Persistance	Sauvegarde IndexedDB automatique

Gestion des Tâches

```
interface Task {
  id: string;
  title: string;
  description?: string;
  columnId: string;
  priority: 'low' | 'medium' | 'high';
  tags: string[];
  subtasks: Subtask[];
  comments: Comment[];
  dueDate?: number;
  createdAt: number;
  updatedAt?: number;
}
```

Fonctionnalité	Détail Technique
Drag & Drop	Bibliothèque @dnd-kit avec animations fluides
Priorités	3 niveaux avec code couleur (vert/jaune/rouge)
Tags	Étiquettes personnalisées avec recherche
Sous-tâches	Checklist avec progression visuelle
Commentaires	Notes horodatées par tâche
Échéances	Date picker + alerte visuelle si dépassée
Recherche	Filtrage en temps réel par titre/tag

2. 🧠 Module Flashcards avec IA Gemini

Intégration Gemini 2.5

```
// Service Gemini - geminiService.ts
const model = genAI.getGenerativeModel({
  model: 'gemini-2.5-flash-preview-05-20'
});

// Prompt engineering pour génération de flashcards
const prompt = `Generate ${count} flashcards about "${topic}"
in JSON format with question and answer fields...`;
```

Fonctionnalité	Détail Technique
Génération IA	Appel API Gemini avec prompt optimisé
Parsing JSON	Extraction robuste des cartes du markdown
Decks	Organisation par matière/thème
Mode Révision	Interface flip-card avec animation CSS 3D
Export PDF	Génération via jsPDF avec mise en page
CRUD complet	Création, édition, suppression de cartes

Structure des Données

```
interface FlashcardDeck {
  id: string;
  title: string;
  description?: string;
  cards: Flashcard[];
  createdAt: number;
  lastStudied?: number;
}

interface Flashcard {
  id: string;
  question: string;
  answer: string;
  difficulty?: 'easy' | 'medium' | 'hard';
}
```

---

3. 🎮 Système de Gamification

Mécanique XP & Niveaux

```
// Calcul du niveau basé sur l'XP
const calculateLevel = (xp: number): number => {
  return Math.floor(xp / 100) + 1;
};

// Rangs progressifs
const RANKS = ['Bronze', 'Silver', 'Gold', 'Platinum', 'Diamond'];

const getRank = (level: number): string => {
  if (level >= 50) return 'Diamond';
  if (level >= 30) return 'Platinum';
  if (level >= 15) return 'Gold';
  if (level >= 5) return 'Silver';
  return 'Bronze';
};
```

Fonctionnalité	Récompense XP
Compléter une tâche	+10 XP
Terminer une sous-tâche	+5 XP
Session Pomodoro complète	+15 XP
Révision de 10 flashcards	+20 XP
Streak quotidien	+5 XP × jours

Interface HUD

- **Barre de progression :** XP vers prochain niveau
- **Badge de rang :** Affiché avec couleur distinctive
- **Statistiques :** Tâches complétées, temps focus, cartes révisées
- **Objectifs quotidiens :** Goals personnalisables

---

4. ⏱️ Focus Timer (Pomodoro)

Configuration

```
const POMODORO_DEFAULTS = {
  focusDuration: 25 * 60,        // 25 minutes
  shortBreak: 5 * 60,            // 5 minutes
  longBreak: 15 * 60,            // 15 minutes
  sessionsBeforeLongBreak: 4
};
```

Fonctionnalité	Détail

Fonctionnalité	Détail
Timer circulaire	Affichage visuel avec animation
Sons ambiants	3 pistes audio (Lo-Fi, Nature, White Noise)
Notifications	Alerte sonore + visuelle en fin de session
Auto-switch	Passage automatique focus ↔ pause
Statistiques	Compteur de sessions complétées

---

5. 🎨 Système de Thèmes

Architecture

```
// ThemeContext.ts
type ThemeMode = 'galilee' | 'pro';
type Language = 'fr' | 'en';

interface ThemeContextType {
  theme: ThemeMode;
  toggleTheme: () => void;
  language: Language;
  setLanguage: (lang: Language) => void;
  t: (key: string) => string;  // Fonction de traduction
}
```

Thème Galilée (Sci-Fi)

- **Palette :** Cyan (#22d3ee), Slate foncé, accents néon
- **Background 3D :** Particules animées avec Three.js
- **Typographie :** Font tech "Rajdhani" + "JetBrains Mono"
- **Effets :** Glassmorphism, ombres néon, animations

Thème Professionnel

- **Palette :** Bleu sobre, gris, blanc
- **Background :** Couleur unie ou gradient léger
- **Typographie :** "Inter" pour lisibilité
- **Style :** Minimaliste, focus sur le contenu

---

6. ⌨️ Palette de Commandes (Cmd+K)

Inspirée de VS Code et Raycast :

```
const commands = [
  { id: 'new-task', label: 'Nouvelle tâche', shortcut: 'N' },
  { id: 'new-column', label: 'Nouvelle colonne', shortcut: 'C' },
  { id: 'flashcards', label: 'Ouvrir Flashcards', shortcut: 'F' },
  { id: 'timer', label: 'Focus Timer', shortcut: 'T' },
  { id: 'theme', label: 'Changer de thème', shortcut: 'Shift+T' },
  { id: 'search', label: 'Rechercher', shortcut: '/' },
];
```

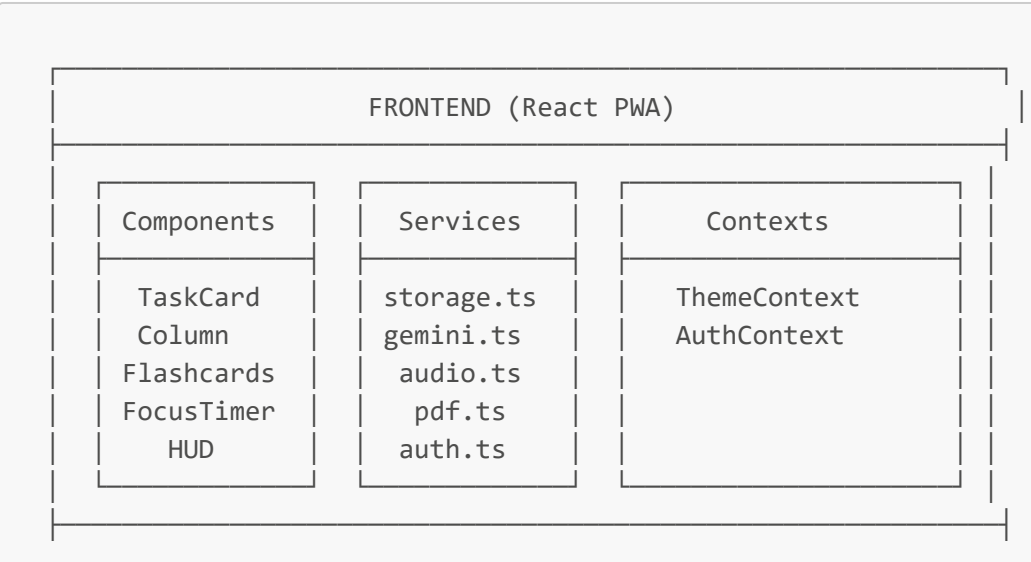
---

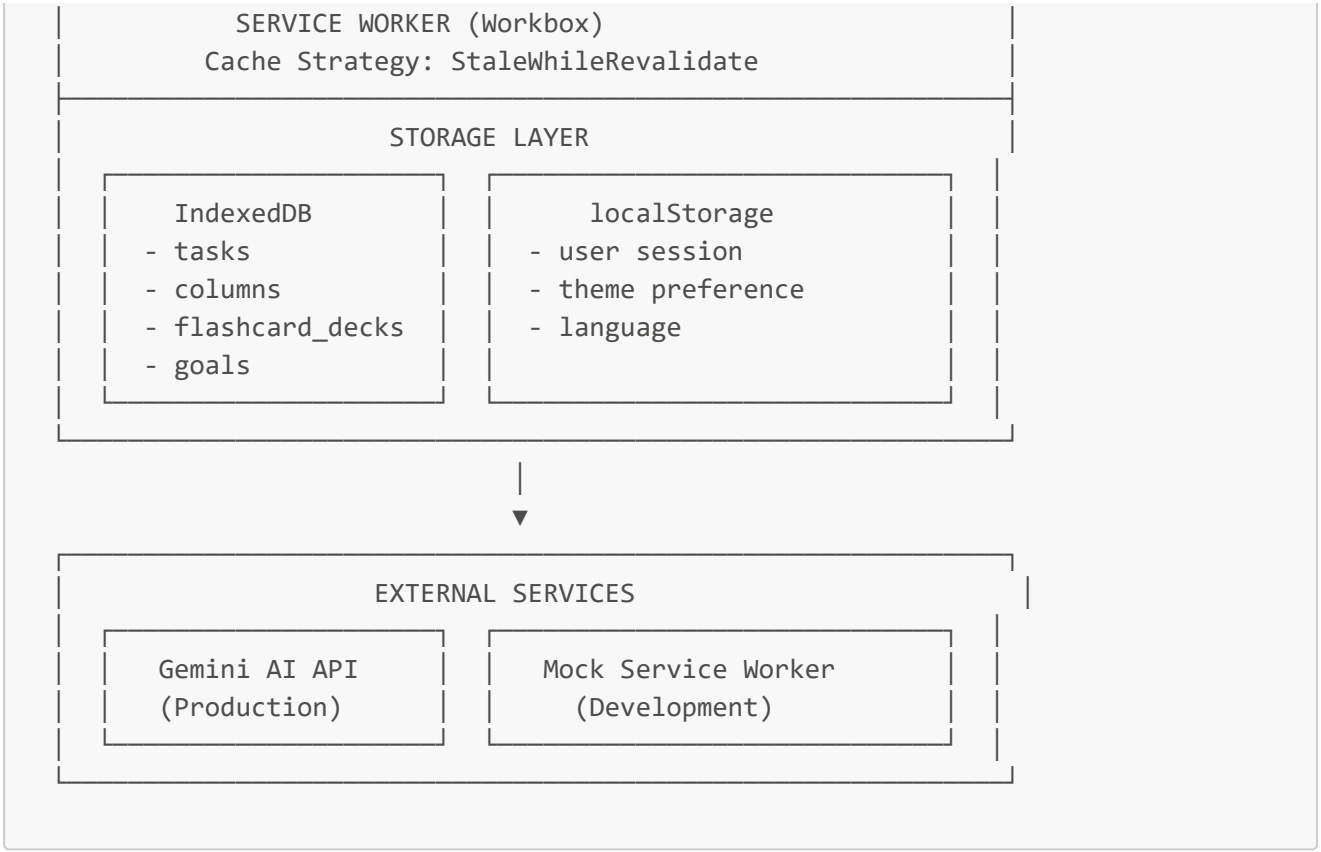
7. 🔊 Service Audio

```
// audioService.ts
class AudioManager {
  private sounds: Map<string, HTMLAudioElement>;

  play(sound: SoundType, theme: ThemeMode): void;
  setVolume(volume: number): void;
  stopAll(): void;
}

type SoundType =
  | 'click'
  | 'success'
  | 'levelup'
  | 'drop'
  | 'ambient-lofi'
  | 'ambient-nature';
```

🏗️ Architecture Technique

Vue d'Ensemble





Structure des Fichiers

```
Galilee-OS/
├── frontend/
│   ├── src/
│   │   ├── components/          # Composants React
│   │   │   ├── App.tsx
│   │   │   ├── Column.tsx
│   │   │   ├── TaskCard.tsx
│   │   │   ├── EditModal.tsx
│   │   │   ├── FlashcardModule.tsx
│   │   │   ├── FocusTimer.tsx
│   │   │   ├── GamificationHUD.tsx
│   │   │   ├── CommandPalette.tsx
│   │   │   ├── LandingPage.tsx
│   │   │   ├── Login.tsx
│   │   │   ├── Background3D.tsx
│   │   │   ├── ThemeContext.tsx
│   │   │   ├── AuthContext.tsx
│   │   │   └── ui/              # Composants UI réutilisables
│   │   │
│   │   ├── services/            # Logique métier
│   │   │   ├── storage.ts
│   │   │   ├── geminiService.ts
│   │   │   ├── audioService.ts
│   │   │   ├── pdfService.ts
│   │   │   ├── auth.ts
│   │   │   └── db.ts
```

```
│   │   │
│   │   ├── hooks/              # Custom hooks
│   │   ├── contexts/           # Contextes React
│   │   ├── types/              # Types TypeScript
│   │   ├── utils/              # Utilitaires
│   │   └── mocks/              # MSW handlers
│   │
│   ├── public/
│   │   ├── manifest.json
│   │   ├── sw.js
│   │   └── icons/
│   │
│   ├── tests/e2e/              # Tests Playwright
│   ├── package.json
│   ├── vite.config.ts
│   ├── tailwind.config.js
│   └── playwright.config.ts
│
├── .github/workflows/
│   └── ci.yml                  # Pipeline CI/CD
│
├── docker-compose.yml
└── README.md
```

🛠️ Stack Technologique

Frontend

Technologie	Version	Rôle	Justification
React	19.2	Framework UI	Composants réactifs, hooks, écosystème riche
TypeScript	5.8	Typage	Sécurité du code, autocomplétion, refactoring
Vite	7.2	Build tool	HMR ultra-rapide, ESM natif, config minimale
Tailwind CSS	4.1	Styling	Utility-first, responsive, customisable
@dnd-kit	6.3	Drag & Drop	Accessible, performant, flexible
Three.js	0.181	3D Graphics	Background animé thème Galilée
Framer Motion	12.x	Animations	Transitions fluides, gestures
Lucide React	0.556	Icônes	Icônes SVG modernes et légères
jsPDF	2.5	Export PDF	Génération côté client

Services & APIs

Service	Utilisation	Configuration

Service	Utilisation	Configuration
Gemini AI	Génération flashcards	`VITE_GEMINI_API_KEY`
IndexedDB	Stockage offline	Via idb library
localStorage	Session utilisateur	API native
Web Audio API	Sons et musique	Intégré au navigateur

DevOps & Testing

Outil	Version	Rôle
Vitest	4.0	Tests unitaires
Playwright	1.52	Tests E2E cross-browser
ESLint	9.x	Linting + React rules
Docker	-	Conteneurisation
GitHub Actions	-	CI/CD automatisé

PWA Technologies

Technologie	Implémentation
Service Worker	Workbox pour cache intelligent
Web App Manifest	manifest.json complet
IndexedDB	4 stores (tasks, columns, decks, goals)
Cache API	Assets statiques précachés

📦 Installation & Configuration

Prérequis

Outil	Version Minimum	Vérification
Node.js	20.x	`node --version`
npm	10.x	`npm --version`
Git	2.x	`git --version`
Docker	24.x	`docker --version` (optionnel)

Installation Rapide

```
# 1. Cloner le repository
git clone https://github.com/KESHRUD/Galilee-OS.git
cd Galilee-OS

# 2. Installer les dépendances frontend
cd frontend
npm install

# 3. Copier le fichier d'environnement
cp .env.example .env

# 4. Configurer la clé API Gemini (optionnel)
# Éditer .env et ajouter : VITE_GEMINI_API_KEY=votre_cle

# 5. Lancer en développement
npm run dev
```

Variables d'Environnement

frontend/.env

```
# API Gemini pour les flashcards IA
# Obtenir une clé : https://aistudio.google.com/apikey
VITE_GEMINI_API_KEY=AIza...

# Note : L'app fonctionne sans clé Gemini
# Les flashcards IA seront simplement désactivées
```

Installation avec Docker

```
# Build et lancement
docker-compose up -d

# Vérifier les conteneurs
docker-compose ps

# Accéder à l'application
open http://localhost:80
```

📖 Guide d'Utilisation

Premier Lancement

1. **Inscription :** Créez un compte avec nom d'utilisateur + spécialité
2. **Tutoriel :** La landing page présente les fonctionnalités

3. **Dashboard :** Accédez au tableau Kanban principal

Raccourcis Clavier

Raccourci	Action
`Cmd/Ctrl + K`	Ouvrir palette de commandes
`N`	Nouvelle tâche
`C`	Nouvelle colonne
`F`	Ouvrir Flashcards
`T`	Ouvrir Focus Timer
`Shift + T`	Changer de thème
`/`	Rechercher
`Escape`	Fermer modal/palette

Mode Mock (Développement)

Pour tester sans clé API Gemini :

```
http://localhost:5173?msw=on
```

Le Mock Service Worker simulera les réponses API.

🧪 Tests & Qualité

Exécution des Tests

```
cd frontend

# Tests unitaires
npm run test              # Mode watch
npm run test -- --run     # Single run

# Tests E2E
npm run test:e2e          # Headless
npx playwright test --ui  # Mode interactif

# Linting
npm run lint
npm run lint:fix
```

Couverture des Tests

Type	Fichiers	Tests	Statut
Unitaires	4	26	☑ Pass
E2E	2	17	☑ Pass
Total	**6**	**43**	☑

Tests E2E - Navigateurs

Navigateur	Version	Statut
Chromium	Latest	☑
Firefox	Latest	☑
WebKit	Latest	☑
Mobile Chrome	Emulated	☑
Mobile Safari	Emulated	☑

Scénarios Testés

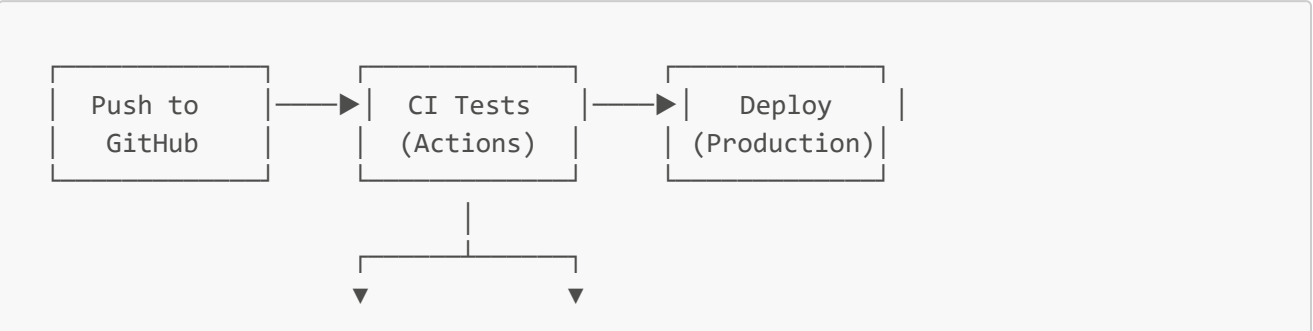
- ☑ Création/édition/suppression de tâches
- ☑ Drag & drop entre colonnes
- ☑ Génération de flashcards
- ☑ Timer Pomodoro
- ☑ Changement de thème
- ☑ Mode offline
- ☑ Installation PWA
- ☑ Service Worker registration

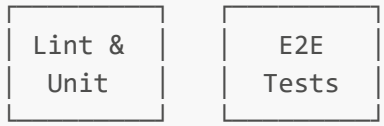
🚀 Déploiement

Production

URL Production : https://galilee-os.netlify.app

Pipeline CI/CD





☑ Conformité PWA

Checklist des Critères

Critère	Implémentation	Statut
HTTPS	Netlify HTTPS auto	☑
Manifest valide	manifest.json complet	☑
Service Worker	Workbox + cache strategies	☑
Icônes	192x192 + 512x512 SVG	☑
Installable	Prompt d'installation	☑
Offline capable	IndexedDB + cache	☑
Responsive	Mobile-first Tailwind	☑
Performance	Vite optimizations	☑
Accessibilité	ARIA labels, keyboard nav	☑

Manifest.json

```
{
  "name": "Galilée OS",
  "short_name": "Galilée",
  "description": "PWA de productivité pour étudiants",
  "start_url": "/",
  "display": "standalone",
  "background_color": "#0f172a",
  "theme_color": "#22d3ee",
  "icons": [
    { "src": "/icons/icon-192x192.svg", "sizes": "192x192" },
    { "src": "/icons/icon-512x512.svg", "sizes": "512x512" }
  ]
}
```

🗺 Roadmap

Version 1.0 (Actuelle) ☑

- ☑ Tableau Kanban complet
- ☑ Flashcards avec Gemini AI
- ☑ Gamification XP/Niveaux
- ☑ Focus Timer Pomodoro
- ☑ Thèmes Galilée/Pro
- ☑ PWA offline-first
- ☑ Tests E2E complets

Version 1.1 (Planifiée)

- ⏳ Synchronisation cloud (Firebase)
- ⏳ Collaboration temps réel
- ⏳ Mode examen (flashcards chronométrées)
- ⏳ Statistiques avancées
- ⏳ Notifications push

Version 2.0 (Future)

- 📆 Application mobile native (React Native)
- 📆 Intégration calendrier universitaire
- 📆 API publique
- 📆 Marketplace de decks flashcards

📄 Licence

Ce projet est sous licence **MIT** - voir le fichier LICENSE pour plus de détails.

🙏 Remerciements

- **Sup Galilée** - Pour le cadre académique et les ressources
- **Université Sorbonne Paris Nord** - Institution d'accueil
- **Google AI** - Pour l'API Gemini
- **La communauté Open Source** - Pour les outils incroyables

---

Développé par

BENHAMMADA Ahmed Amine
Étudiant Ingénieur Informatique - ING2

GITHUB **KESHRUD**

Université Sorbonne Paris Nord
Sup Galilée - École d'Ingénieurs
Programmation Web Avancée - 2024/2025

Star ce repo si le projet vous a été utile !

Made with , React, et beaucoup de