

AgroGest PWA - Aplicação de Gestão Agrícola


Uma Progressive Web App (PWA) moderna e elegante para gestão de atividades agrícolas, desenvolvida com React + TypeScript e design inspirado no iOS 17.



Funcionalidades



Planeamento de Tarefas

- Criação e gestão de tarefas agrícolas (plantio, rega, pulverização, colheita)
- Organização por campos e datas
- Estados: pendente, em curso, concluída
- Interface intuitiva com cartões visuais
-  **Registo por Voz:** Adicione tarefas usando comandos de voz em português



Gestão de Atividades

- Visualização completa de todas as atividades
- Filtros por tipo, status e pesquisa
- Agrupamento por período (hoje, esta semana, futuro)
- Estatísticas de produtividade




Controlo de Inventário

- Gestão de insumos agrícolas (sementes, fertilizantes, pesticidas, combustível)
- Alertas de stock baixo
- Controlo de entradas e saídas
- Cálculo automático do valor total



Dashboard Analítico

- KPIs principais (área tratada, água utilizada, fertilizante, custos)
- Gráficos de evolução temporal
- Alertas de inventário
- Métricas de produtividade

-  **Reconhecimento de Imagem:** Capture fotos de plantações para identificar pragas e doenças

Definições

- Gestão de perfil do utilizador
- Sincronização de dados
- Backup e restauro
- Informações da aplicação

Design & UX

Estética iOS 17

- **Fundo Global:** Imagem fixa de campos de milho em alta resolução
- **Overlay Universal:** Verde-escuro (#013220) com efeito blur
- **Tipografia:** SF Pro Text/Display com fallback system-ui
- **Cantos Arredondados:** 12px em todos os elementos
- **Tab Bar Inferior:** Navegação com 5 separadores

Cores da Marca

- **Verde Principal:** #013220 (tema principal)
- **Verde Secundário:** #064e3b
- **Laranja:** #ea580c (do logótipo)
- **Verde Claro:** #d1fae5

🔧 Stack Tecnológico

Frontend

- **React 18 + TypeScript** - Interface de utilizador moderna
- **Vite** - Build tool rápido e otimizado
- **Tailwind CSS** - Framework CSS utilitário
- **React Router** - Navegação SPA
- **Storybook** - Desenvolvimento e documentação de componentes

Estado & Dados

- **Zustand** - Gestão de estado leve e performante
- **IndexedDB** - Armazenamento local offline-first
- **PWA** - Service Worker para funcionalidade offline
- **Firebase** - Backend serverless para sincronização de dados

PWA & Mobile

- **Capacitor** - Recursos nativos (câmara, microfone)
- **Service Worker** - Cache offline e atualizações
- **Web App Manifest** - Instalação em dispositivos



Instalação & Uso

Requisitos

- Node.js 18+
- pnpm (recomendado) ou npm

Desenvolvimento Local

```
# Clonar o repositório
git clone <repository-url>
cd agrogest-pwa

# Instalar dependências
pnpm install

# Iniciar servidor de desenvolvimento
pnpm dev

# Aceder a http://localhost:3000
```

Build de Produção

```
# Construir aplicação
pnpm build

# Servir build localmente para teste
pnpm preview
```

Scripts Disponíveis

pnpm dev	# Servidor de desenvolvimento
pnpm build	# Build de produção
pnpm preview	# Servir build localmente
pnpm lint	# Verificar código
pnpm type-check	# Verificar tipos TypeScript

Estrutura do Projeto

```
src/
├── components/           # Componentes reutilizáveis
│   ├── ui/              # Componentes base (Card, Button, Input)
│   ├── layout/          # Layout e navegação (Header, TabBar)
│   └── shared/           # Componentes compartilhados
├── pages/               # Páginas da aplicação
│   ├── HojePage.tsx      # Página principal
│   ├── AtividadesPage.tsx # Gestão de atividades
│   ├── InventarioPage.tsx # Gestão de inventário
│   ├── DashboardPage.tsx # Analytics
│   └── DefinicoesPage.tsx # Configurações
├── hooks/               # Custom hooks
├── services/            # APIs e serviços externos
├── stores/              # Estado Zustand
│   ├── useAppStore.ts    # Estado global da app
│   ├── useTaskStore.ts   # Gestão de tarefas
│   └── useInventoryStore.ts # Gestão de inventário
├── types/               # Tipos TypeScript
├── utils/               # Utilitários
└── assets/              # Recursos estáticos
```

Modelo de Dados

Tarefas Agrícolas






```
interface Task {  
  id: string  
  titulo: string  
  descricao?: string  
  tipo: 'plantio' | 'rega' | 'pulverizacao' | 'colheita'  
  campo: string  
  dataExecucao: Date  
  horaExecucao: string  
  status: 'pendente' | 'em_curso' | 'concluida'  
  userId: string  
}
```

Itens de Inventário

```
interface InventoryItem {  
  id: string  
  nome: string  
  categoria: 'sementes' | 'fertilizantes' | 'pesticidas' |  
  'combustivel'  
  quantidade: number  
  unidade: string  
  precoUnitario: number  
  fornecedor?: string  
  dataVencimento?: Date  
  stockMinimo: number  
  userId: string  
}
```


PWA Features

Características PWA

-  **Instalável** - Pode ser instalada como app nativa
-  **Offline-First** - Funciona sem internet
-  **Responsiva** - Adapta-se a todos os dispositivos
-  **Rápida** - Carregamento otimizado
-  **Segura** - HTTPS obrigatório

Service Worker

- Cache automático de recursos
- Sincronização em background
- Atualizações offline

Web App Manifest

```
{  
  "name": "AgroGest PWA",  
  "short_name": "AgroGest",  
  "description": "Aplicação de gestão agrícola",  
  "theme_color": "#013220",  
  "background_color": "#013220",  
  "display": "standalone"  
}
```

Configuração de Ambiente

Variáveis de Ambiente (.env)

```
# Firebase (Opcional)
VITE_FIREBASE_API_KEY=your_api_key
VITE_FIREBASE_AUTH_DOMAIN=your_domain
VITE_FIREBASE_PROJECT_ID=your_project_id

# Google APIs (Opcional)
VITE_GOOGLE_MAPS_API_KEY=your_maps_key
VITE_GOOGLE_SPEECH_API_KEY=your_speech_key
```

Deploy

Netlify/Vercel

```
# Build da aplicação
pnpm build

# Upload da pasta 'dist' para o serviço
```

Storybook

```
# Iniciar Storybook localmente
pnpm storybook

# Build do Storybook
pnpm build-storybook
```

Docker

```
FROM node:18-alpine
WORKDIR /app
COPY package*.json ./
RUN npm ci --only=production
COPY . .
RUN npm run build
EXPOSE 3000
CMD ["npm", "run", "preview"]
```

Testes

Testes Unitários (Vitest)




```
pnpm test
```

Testes E2E

```
pnpm test:e2e
```

Performance

Métricas de Qualidade

-  **Lighthouse Score:** 95+
-  **First Contentful Paint:** < 1.5s
-  **Time to Interactive:** < 2s

-  **Cumulative Layout Shift:** < 0.1

Otimizações Implementadas

- Code splitting automático
- Lazy loading de componentes
- Imagens otimizadas
- Cache eficiente
- Bundle size otimizado



Segurança

- HTTPS obrigatório
- Sanitização de inputs
- Validação de dados
- Storage seguro local



Acessibilidade

- WCAG 2.1 AA compliance
- Navegação por teclado
- Screen reader friendly
- Contraste adequado (4.5:1)
- Fonte mínima 16px



Contribuição

1. Fork o projeto
2. Crie uma branch para a feature (`git checkout -b feature/AmazingFeature`)

3. Commit suas mudanças (`git commit -m 'Add some AmazingFeature'`)
4. Push para a branch (`git push origin feature/AmazingFeature`)
5. Abra um Pull Request

Licença


Este projeto está licenciado sob a licença MIT - veja o arquivo [LICENSE](#) para detalhes.

Equipa

- **Desenvolvimento:** MiniMax Agent
- **Design:** Inspirado em iOS 17
- **Logo:** AgroGest Official

Suporte

Para suporte técnico ou dúvidas:

-  Email: suporte@agrogest.pt
-  Website: agrogest.pt
-  Demo: <https://i2gsjz5oq.space.minimax.io>

AgroGest PWA - Modernizando a agricultura portuguesa com tecnologia de ponta!

