```
import React, { useState, useMemo } from 'react'
import { useTaskStore } from '@/stores/useTaskStore'
import { useInventoryStore } from '@/stores/useInventoryStore'
import Card from '@/components/ui/Card'
import KPICard from '@/components/ui/KPICard'
import { KPIData } from '@/types'
import { format, subDays, isAfter, isBefore } from 'date-fns'
import { pt } from 'date-fns/locale/pt'

export const DashboardPage: React.FC = () => {
const { tasks } = useTaskStore()
const { items, getTotalValue } = useInventoryStore()
const [selectedPeriod, setSelectedPeriod] = useState<'7d' | '30d' | '90d'>('30d')

// Calcular KPIs baseados nas tarefas e inventário
const kpiData = useMemo((): KPIData => {
const now = new Date()
const periodDays = selectedPeriod === '7d' ? 7 : selectedPeriod === '30d' ? 30 : 90
const startDate = subDays(now, periodDays)
const recentTasks = tasks.filter(task =>
  isAfter(task.dataExecucao, startDate) &&
  isBefore(task.dataExecucao, now) &&
  task.status === 'concluida'
)

// Simulação de dados baseados nas tarefas
const areaTratada = recentTasks.length * 2.5 // 2.5 ha por tarefa
const litrosAgua = recentTasks.filter(t => t.tipo ===
'rega').length * 150 // 150L por rega
const kgFertilizante = recentTasks.filter(t => t.tipo ===
'pulverizacao').length * 25 // 25kg por pulverização
const custoDiario = (getTotalValue() / periodDays) || 45.50 //
Valor médio

return {
  areaTratada,
  litrosAgua,
  kgFertilizante,
  custoDiario,
  data: now
}
```

}, [tasks, selectedPeriod, getTotalValue])

// Estatísticas das tarefas
const taskStats = useMemo(() => {
const totalTasks = tasks.length
const completedTasks = tasks.filter(t => t.status === 'concluida').length
const pendingTasks = tasks.filter(t => t.status === 'pendente').length
const completionRate = totalTasks > 0 ? (completedTasks / totalTasks) * 100 : 0

```
const tasksByType = {
  plantio: tasks.filter(t => t.tipo === 'plantio').length,
  rega: tasks.filter(t => t.tipo === 'rega').length,
  pulverizacao: tasks.filter(t => t.tipo ===
'pulverizacao').length,
  colheita: tasks.filter(t => t.tipo === 'colheita').length
}

return {
  totalTasks,
  completedTasks,
  pendingTasks,
  completionRate,
  tasksByType
}
```

}, [tasks])

// Alertas de inventário
const inventoryAlerts = useMemo(() => {
const lowStockItems = items.filter(item => item.quantidade <= item.stockMinimo)
const expiringSoon = items.filter(item => {
if (!item.dataVencimento) return false
const dataVencimento = item.dataVencimento instanceof Date ? item.dataVencimento : new Date(item.dataVencimento)
const daysUntilExpiry = Math.ceil(
(dataVencimento.getTime() - Date.now()) / (1000 * 60 * 60 * 24)
)
return daysUntilExpiry <= 30 && daysUntilExpiry >= 0
})

```
return { lowStockItems, expiringSoon }
```

}, [items])

const periods = [
{ value: '7d', label: '7 dias' },
{ value: '30d', label: '30 dias' },
{ value: '90d', label: '90 dias' }
]

return (

{/ Seletor de período /}

Dashboard

{periods.map(period => (
`setSelectedPeriod(period.value as any)}`

```
        className={ px-3 py-1 rounded-lg text-sm font-medium transition-all ${
                    selectedPeriod === period.value
                      ? 'bg-white/20 text-white'
                      : 'text-green-100 hover:bg-white/10'
                    } }

                  >

              {period.label}
```

))}

Dados dos últimos {periods.find(p => p.value === selectedPeriod)?.label}

```
{/* KPIs principais */}
<div className="grid grid-cols-2 gap-4">
  <KPICard
    title="Área Tratada"
    value={kpiData.areaTratada.toFixed(1)}
    unit="ha"
    icon={<span className="text-2xl">🌾</span>}
    trend={{
      value: 12.5,
      isPositive: true
    }}
  />
  <KPICard
    title="Água Utilizada"
    value={kpiData.litrosAgua}
    unit="L"
    icon={<span className="text-2xl">💧</span>}
    trend={{
      value: 8.3,
      isPositive: false
    }}
  />
  <KPICard
    title="Fertilizante"
    value={kpiData.kgFertilizante}
    unit="kg"
    icon={<span className="text-2xl">🧪</span>}
    trend={{
      value: 5.7,
      isPositive: true
    }}
  />
  <KPICard
    title="Custo Diário"
    value={kpiData.custoDiario.toFixed(2)}
```

```
      unit="€"
      icon={<span className="text-2xl">💰</span>}
      trend={{
        value: 3.2,
        isPositive: false
      }}
    />
  </div>

  {/* Estatísticas das tarefas */}
  <Card variant="overlay">
    <h3 className="text-lg font-semibold mb-4 flex items-center
gap-2">
      <span>📊</span>
      Produtividade
    </h3>

    <div className="space-y-4">
      {/* Taxa de conclusão */}
      <div>
        <div className="flex justify-between items-center mb-2">
          <span className="text-sm text-green-100">Taxa de
Conclusão</span>
          <span className="text-sm font-medium">
            {taskStats.completionRate.toFixed(1)}%
          </span>
        </div>
        <div className="w-full bg-green-800/50 rounded-full h-2">
          <div

className="bg-green-400 h-2 rounded-full transition-all
duration-300"
            style={{ width: `${taskStats.completionRate}%` }}
          />
        </div>
      </div>

      {/* Distribuição por tipo */}
      <div>
        <h4 className="text-sm font-medium mb-3 text-green-100">
          Atividades por Tipo
        </h4>
        <div className="grid grid-cols-2 gap-3">
          <div className="flex items-center justify-between">
            <span className="text-sm">🌱 Plantio</span>
            <span className="font-
medium">{taskStats.tasksByType.plantio}</span>
          </div>
          <div className="flex items-center justify-between">
            <span className="text-sm">💧 Rega</span>
            <span className="font-
medium">{taskStats.tasksByType.rega}</span>
          </div>
          <div className="flex items-center justify-between">
            <span className="text-sm">🚿 Pulverização</span>
            <span className="font-
medium">{taskStats.tasksByType.pulverizacao}</span>
          </div>
          <div className="flex items-center justify-between">
            <span className="text-sm">🌾 Colheita</span>
            <span className="font-
medium">{taskStats.tasksByType.colheita}</span>
          </div>
        </div>
      </div>
    </div>
  </Card>

  {/* Alertas */}
  {(inventoryAlerts.lowStockItems.length > 0 ||
inventoryAlerts.expiringSoon.length > 0) && (
    <Card variant="overlay" className="border-yellow-500/50">
      <h3 className="text-lg font-semibold mb-4 flex items-center
gap-2">
        <span>⚠️</span>
        Alertas
      </h3>

      <div className="space-y-3">
        {inventoryAlerts.lowStockItems.length > 0 && (
          <div className="bg-red-500/20 rounded-lg p-3">
            <h4 className="font-medium text-red-200 mb-2">
              Stock Baixo ({inventoryAlerts.lowStockItems.length}
itens)
            </h4>
            <div className="space-y-1">
              {inventoryAlerts.lowStockItems.slice(0, 3).map(item
=> (
                <div key={item.id} className="text-sm text-
red-100">
                  {item.nome}: {item.quantidade} {item.unidade}
                </div>
              ))}
              {inventoryAlerts.lowStockItems.length > 3 && (
                <div className="text-sm text-red-200">
                  ... e mais {inventoryAlerts.lowStockItems.length
- 3} itens
                </div>
              )}
            </div>
          </div>
        )}

        {inventoryAlerts.expiringSoon.length > 0 && (
          <div className="bg-yellow-500/20 rounded-lg p-3">
```

```
            <h4 className="font-medium text-yellow-200 mb-2">
              A Vencer em 30 dias
({inventoryAlerts.expiringSoon.length} itens)
            </h4>
            <div className="space-y-1">
              {inventoryAlerts.expiringSoon.slice(0, 3).map(item
=> (
                <div key={item.id} className="text-sm text-
yellow-100">
                  {item.nome}: {item.dataVencimento ?
(item.dataVencimento instanceof Date ? item.dataVencimento : new
Date(item.dataVencimento)).toLocaleDateString('pt-PT') : ''}
                </div>
              ))}
            </div>
          </div>
        )}
      </div>
    </Card>
  )}

  {/* Gráfico simples de evolução */}
  <Card variant="overlay">
    <h3 className="text-lg font-semibold mb-4 flex items-center
gap-2">
      <span>📈</span>
      Evolução Semanal
    </h3>

    {/* Gráfico simplificado */}
    <div className="space-y-4">
      {['Seg', 'Ter', 'Qua', 'Qui', 'Sex', 'Sáb', 'Dom'].map((day,
index) => {
        const value = Math.random() * 100 // Dados simulados
        return (
          <div key={day} className="flex items-center gap-3">
            <span className="text-sm w-8 text-green-100">{day}</
span>
            <div className="flex-1 bg-green-800/50 rounded-full
h-2">
              <div
                className="bg-green-400 h-2 rounded-full
transition-all duration-300"
                style={{ width: `${value}%` }}
              />
            </div>
            <span className="text-sm w-10 text-right font-medium">
              {value.toFixed(0)}%
            </span>
          </div>
        )
      })}
    </div>
  </Card>
</div>
```

)
}

export default DashboardPage