📋 ESTRUTURA COMPLETA DO PROJETO

Sistema de Inventário de Suplementos

Projeto: Sistema Web de Inventário de Suplementos
Tecnologia Principal: React.js + Firebase
URL: https://inventario-8b388.web.app

🗂️ VISÃO GERAL DA ESTRUTURA

```
inventario/
├── 📁 public/                    # Arquivos públicos estáticos
│   ├── index.html               # HTML principal
│   ├── favicon.ico              # Ícone do site
│   └── manifest.json            # Configuração PWA
│
├── 📁 src/                      # Código fonte principal
│   ├── 📁 components/           # Componentes reutilizáveis
│   │   ├── Header.js           # Barra de navegação
│   │   ├── Login.js            # Tela de autenticação
│   │   ├── ProtectedRoute.js   # Proteção de rotas
│   │   └── SupplementForm.js   # Formulário de suplementos
│   │
│   ├── 📁 contexts/             # Gerenciamento de estado
│   │   ├── AuthContext.js      # Contexto de autenticação
│   │   └── InventoryContext.js # Contexto do inventário
│   │
│   ├── 📁 pages/                # Páginas principais
│   │   ├── Dashboard.js        # Painel de controle
│   │   ├── Inventory.js        # Gerenciamento de estoque
│   │   └── Reports.js          # Sistema de relatórios
│   │
│   ├── 📁 services/             # Serviços externos
│   │   └── firebase.js         # Configuração Firebase
│   │
│   ├── App.js                  # Componente principal
│   └── index.js                # Ponto de entrada
│
├── 📁 .firebase/                # Cache do Firebase (ignorado)
├── 📁 .github/                  # Configurações GitHub
├── 📁 node_modules/             # Dependências (ignorado)
│
├── 📄 package.json              # Configuração do projeto
├── 📄 firebase.json             # Configuração Firebase
├── 📄 .firebaserc              # Projeto Firebase ativo
├── 📄 .gitignore               # Arquivos ignorados pelo Git
├── 📄 README.md                # Documentação principal
└── 📄 DOCUMENTACAO_COMPLETA.md # Documentação técnica
```

📂 DETALHAMENTO POR DIRETÓRIO

📁 `src/components/` - Componentes Reutilizáveis

🧭 `Header.js` - Barra de Navegação

```
Localização: src/components/Header.js
Função: Barra de navegação principal do sistema
```

Responsabilidades:

- Exibir menu de navegação (Dashboard, Inventário, Relatórios)
- Mostrar nome do usuário logado
- Botão de logout com confirmação
- Logo e título do sistema
- Responsividade mobile/desktop

Componentes Material-UI utilizados:

- `AppBar` - Barra superior
- `Toolbar` - Container do menu
- `Button` - Botões de navegação
- `IconButton` - Botão de logout
- `Typography` - Texto e títulos

Estado gerenciado:

- Navegação ativa atual
- Status do menu mobile
- Dados do usuário logado

🔐 `Login.js` - Tela de Autenticação

```
Localização: src/components/Login.js
Função: Interface de login com Google OAuth
```

Responsabilidades:

- Exibir formulário de login
- Integração com Google Sign-In
- Feedback visual durante autenticação
- Tratamento de erros de login
- Redirecionamento pós-login

Fluxo de autenticação:

1. Usuário clica em "Entrar com Google"
2. Popup do Google OAuth abre
3. Usuário autoriza a aplicação
4. Token retorna para o Firebase
5. Contexto de autenticação é atualizado
6. Redirecionamento para Dashboard

Estados possíveis:

- `idle` - Aguardando ação do usuário
- `loading` - Processando login
- `error` - Erro na autenticação
- `success` - Login realizado com sucesso

`ProtectedRoute.js` - Proteção de Rotas

```
Localização: src/components/ProtectedRoute.js
Função: Proteger páginas que requerem autenticação
```

Responsabilidades:

- Verificar se usuário está autenticado
- Redirecionar para login se não autenticado
- Permitir acesso a conteúdo protegido
- Exibir loading durante verificação

Lógica de proteção:

```
if (loading) return <LoadingSpinner />;
if (!user) return <Navigate to="/login" />;
return children; // Renderiza o conteúdo protegido
```

Páginas protegidas:

- `/` - Dashboard
- `/inventory` - Inventário
- `/reports` - Relatórios

`SupplementForm.js` - Formulário de Suplementos

```
Localização: src/components/SupplementForm.js
Função: Formulário para adicionar/editar suplementos
```

Campos do formulário:

- **Nome** (obrigatório)
- **Categoria** (dropdown)
- **Marca** (obrigatório)
- **Quantidade** (número, min: 0)
- **Unidade** (dropdown: kg, g, ml, unidades, etc.)
- **Preço de Compra** (moeda, obrigatório)
- **Preço de Venda** (moeda, obrigatório)
- **Data de Compra** (date picker)
- **Data de Vencimento** (date picker)
- **Fornecedor** (texto)
- **Descrição** (textarea)
- **Estoque Mínimo** (número, padrão: 1)

Validações implementadas:

- Campos obrigatórios
- Valores numéricos positivos
- Data de vencimento posterior à compra
- Preço de venda maior que compra (aviso)
- Formato de moeda brasileiro

Tecnologias utilizadas:

- React Hook Form para gerenciamento
- Material-UI para componentes visuais
- Validação em tempo real
- Máscaras para campos de moeda

🗀 `src/contexts/` - Gerenciamento de Estado Global

👤 `AuthContext.js` - Contexto de Autenticação

```
Localização: src/contexts/AuthContext.js
Função: Gerenciar estado de autenticação global
```

Estados fornecidos:

```
{
  user: null | UserObject,      // Dados do usuário logado
  loading: boolean,             // Status de carregamento
  signInWithGoogle: function,   // Função de login
  logout: function              // Função de logout
}
```

Dados do usuário incluem:

- `uid` - ID único do Firebase
- `displayName` - Nome completo
- `email` - Email do Google
- `photoURL` - Foto do perfil
- `providerId` - Provedor (google.com)

Funcionalidades:

- Persistência automática de sessão
- Listener de mudanças de estado
- Logout com limpeza de dados
- Tratamento de erros de autenticação

`InventoryContext.js` - Contexto do Inventário

```
Localização: src/contexts/InventoryContext.js
Função: Gerenciar todos os dados e operações do inventário
```

Estados fornecidos:

```
{
  supplements: Array,                    // Lista de suplementos
  loading: boolean,                      // Status de operações
  error: string | null,                  // Mensagens de erro

  // CRUD Operations
  addSupplement: function,               // Adicionar suplemento
  updateSupplement: function,            // Atualizar suplemento
  deleteSupplement: function,            // Deletar suplemento
  fetchSupplements: function,            // Recarregar lista

  // Análises e Relatórios
  getSupplementsByCategory: function,    // Agrupar por categoria
  getLowStockSupplements: function,      // Produtos com estoque baixo
  getExpiringSoon: function,             // Produtos vencendo

  // Cálculos Financeiros
  getTotalValue: function,               // Valor total de compra
  getTotalSaleValue: function,           // Valor total de venda
  getTotalProfit: function,              // Lucro total potencial
  getProfitMargin: function,             // Margem de lucro %
  getSupplementsWithProfit: function     // Lista com lucros calculados
}
```

Estrutura de dados do suplemento:

```
{
  id: "firebase_doc_id",
  nome: "Whey Protein Gold Standard",
  categoria: "Proteína",
  marca: "Optimum Nutrition",
  quantidade: 2.5,
  unidade: "kg",
  precoCompra: 150.00,
  precoVenda: 200.00,
  dataCompra: "2025-01-15",
  dataVencimento: "2026-01-15",
  fornecedor: "Distribuidora ABC",
  descricao: "Whey concentrado sabor chocolate",
  estoqueMinimo: 1,
  ativo: true,
  userId: "user_firebase_uid",
  createdAt: Date,
  updatedAt: Date,

  // Campos calculados (adicionados dinamicamente)
  unitProfit: 50.00,           // precoVenda - precoCompra
  totalProfit: 125.00,         // unitProfit * quantidade
  profitMargin: 33.33          // (unitProfit / precoCompra) * 100
}
```

Operações CRUD detalhadas:

1. `addSupplement(supplementData)`

- Adiciona `userId` do usuário logado
- Adiciona timestamps (`createdAt`, `updatedAt`)
- Salva no Firestore
- Atualiza estado local
- Retorna o suplemento criado

2. `updateSupplement(id, updateData)`

- Atualiza `updatedAt`
- Atualiza documento no Firestore
- Atualiza estado local
- Mantém dados não alterados

3. `deleteSupplement(id)`

- Remove documento do Firestore
- Remove do estado local
- Operação irreversível

4. `fetchSupplements()`

- Busca todos os suplementos do usuário
- Ordena por nome alfabeticamente
- Atualiza estado local
- Trata erros de conexão

Funções de análise:

1. `getSupplementsByCategory()`

```
// Retorna objeto agrupado por categoria
{
  "Proteína": [supplement1, supplement2],
  "Creatina": [supplement3],
  "Vitaminas": [supplement4, supplement5]
}
```

2. `getLowStockSupplements()`

```
// Retorna array de suplementos onde:
quantidade <= estoqueMinimo
```

3. `getExpiringSoon(days = 30)`

```
// Retorna suplementos vencendo em X dias
dataVencimento <= (hoje + days)
```

Cálculos financeiros implementados:

1. Valor Total de Compra:

```
getTotalValue() {
  return supplements.reduce((total, supplement) => {
    return total + (supplement.precoCompra * supplement.quantidade);
  }, 0);
}
```

2. Valor Total de Venda:

```
getTotalSaleValue() {
  return supplements.reduce((total, supplement) => {
    return total + (supplement.precoVenda * supplement.quantidade);
  }, 0);
}
```

3. Lucro Total:

```
getTotalProfit() {
  return supplements.reduce((total, supplement) => {
    const profit = (supplement.precoVenda - supplement.precoCompra) *
supplement.quantidade;
    return total + profit;
  }, 0);
}
```

4. Margem de Lucro:

```
getProfitMargin() {
  const totalCost = getTotalValue();
  const totalSale = getTotalSaleValue();
  return totalCost > 0 ? ((totalSale - totalCost) / totalCost) * 100 : 0;
}
```

🗀 src/pages/ - Páginas Principais

📊 Dashboard.js - Painel de Controle

```
Localização: src/pages/Dashboard.js
Função: Página principal com estatísticas e alertas
```

Layout da página:

```
┌─────────────────────────────────────────────┐
│                 DASHBOARD                   │
├─────────────────────────────────────────────┤
│  💰 Total     📦 Itens     💵 Venda   📈 Lucro │
│   R$1.500     25 itens    R$2.000   R$500   │
├─────────────────────────────────────────────┤
│              ▨ ALERTAS                      │
│  • 3 produtos com estoque baixo             │
│  • 2 produtos vencendo em 30 dias           │
│  • Margem de lucro: 25%                     │
├─────────────────────────────────────────────┤
│           📊 ANÁLISE POR CATEGORIA          │
│  Proteína: 10 itens                         │
│  Creatina: 5 itens                          │
│  Vitaminas: 8 itens                         │
└─────────────────────────────────────────────┘
```

Cards de estatísticas:

1. **Valor Total de Compra**
   - Soma: `precoCompra × quantidade`
   - Cor: Azul
   - Ícone: 💰
2. **Total de Itens**
   - Contagem de suplementos ativos
   - Cor: Verde
   - Ícone: 📦
3. **Valor Potencial de Venda**
   - Soma: `precoVenda × quantidade`
   - Cor: Roxo
   - Ícone: 💵
4. **Lucro Potencial**
   - Diferença: Venda - Compra
   - Cor: Verde (lucro) / Vermelho (prejuízo)
   - Ícone: 📈
5. **Ticket Médio**
   - Média: `valorTotal / totalItens`
   - Cor: Laranja
   - Ícone: 🎯

Sistema de alertas:

- 🟨 **Estoque Baixo**: `quantidade <= estoqueMinimo`
- 🟧 **Vencimento Próximo**: `dataVencimento <= hoje + 30 dias`
- ⚠️ **Margem Negativa**: `precoVenda < precoCompra`
- 📊 **Análise de Lucratividade**: Margem % colorida

Tecnologias utilizadas:

- Material-UI Grid para layout responsivo
- Cards com elevação e cores temáticas
- Ícones do Material-UI
- Formatação de moeda brasileira
- Hooks para cálculos em tempo real

🏪 `Inventory.js` - Gerenciamento de Estoque

```
Localização: src/pages/Inventory.js
Função: Interface principal para gerenciar suplementos
```

Interface da página:

```
┌─────────────────────────────────────────────────┐
│  🔍 Buscar...    📁 Categoria ▼    ➕ Adicionar │
├─────────────────────────────────────────────────┤
│ Nome        │Categoria│Qtd│Compra│Venda│Ações│
├─────────────────────────────────────────────────┤
│ Whey Gold   │Proteína │2kg│150,00│200,00│ ✏️🗑️ │
│ Creatina    │Creatina │500g│ 80,00│100,00│ ✏️🗑️ │
│ BCAA        │Aminoácido│120│ 45,00│ 60,00│ ✏️🗑️ │
├─────────────────────────────────────────────────┤
│           📄 10 de 250 itens                    │
└─────────────────────────────────────────────────┘
```

Funcionalidades da tabela:

1. **Colunas exibidas:**
   - Nome do produto
   - Categoria
   - Quantidade + Unidade
   - Preço de compra (R$)
   - Preço de venda (R$)
   - Data de vencimento
   - Ações (Editar/Deletar)
2. **Recursos interativos:**
   - **Ordenação**: Click no cabeçalho da coluna
   - **Paginação**: 10, 25, 50 ou 100 itens por página
   - **Busca**: Campo de texto filtra nome/marca
   - **Filtro por categoria**: Dropdown com todas as categorias
   - **Seleção múltipla**: Para ações em lote
3. **Indicadores visuais:**
   - ▨ **Linha amarela**: Estoque baixo
   - ▩ **Linha vermelha**: Vencimento próximo
   - ◍ **Linha verde**: Produto normal
   - ◯ **Linha cinza**: Produto inativo

Ações disponíveis:

- ➕ **Adicionar**: Abre modal com `SupplementForm`
- ✏️ **Editar**: Abre modal com dados preenchidos

- 🗑️ **Deletar**: Confirmação antes de remover
- 👁️ **Visualizar**: Modal com todos os detalhes
- 📊 **Exportar**: CSV com dados selecionados

Estados da página:

- Lista de suplementos carregada
- Filtros aplicados (busca + categoria)
- Modal de formulário (aberto/fechado)
- Suplemento em edição
- Loading durante operações

📈 `Reports.js` - Sistema de Relatórios

```
Localização: src/pages/Reports.js
Função: Geração de relatórios e análises avançadas
```

Interface de relatórios:

```
┌─────────────────────────────────────────────┐
│        📈 RELATÓRIOS E ANÁLISES             │
├─────────────────────────────────────────────┤
│ Categoria: [Todas ▼] Data: [01/01/2025]     │
│ Tipo: [Geral ▼]          🔍 Filtrar         │
├─────────────────────────────────────────────┤
│ Nome     |Cat|Qtd|Compra|Venda|Lucro|Margem │
├─────────────────────────────────────────────┤
│ Whey     |Pro|2kg|300,00|400,00|100,00|33%|  │
│ Creatina |Cre|1kg| 80,00|100,00| 20,00|25%|  │
├─────────────────────────────────────────────┤
│ TOTAIS:        |     |380,00|500,00|120,00|32%|
├─────────────────────────────────────────────┤
│               📥 Exportar CSV               │
└─────────────────────────────────────────────┘
```

Tipos de relatórios:

1. **Relatório Geral**
   - Todos os suplementos
   - Cálculos de lucro incluídos
   - Totalizadores no rodapé
2. **Por Categoria**
   - Filtro por categoria específica
   - Estatísticas da categoria

    - Comparação percentual

3. **Estoque Baixo**
    - Produtos `quantidade <= estoqueMinimo`
    - Sugestões de reposição
    - Priorização por importância

4. **Próximos ao Vencimento**
    - Produtos vencendo em 30/60/90 dias
    - Alertas por urgência
    - Ações recomendadas

Filtros disponíveis:

- **Por categoria**: Dropdown com todas as categorias
- **Por data de compra**: Date picker (a partir de)
- **Por margem de lucro**: Slider de porcentagem
- **Por fornecedor**: Dropdown com fornecedores
- **Por status**: Ativo/Inativo/Todos

Colunas do relatório:

- Nome do produto
- Categoria
- Quantidade + Unidade
- Preço de compra unitário
- Preço de venda unitário
- **Lucro unitário** (calculado)
- **Lucro total** (calculado)
- **Margem de lucro %** (calculada)
- Data de vencimento
- Fornecedor

Funcionalidades de exportação:

```
// Formato do CSV exportado
"Nome,Categoria,Quantidade,Unidade,Preço Compra,Preço Venda,Lucro Unitário,Lucro
Total,Margem %"
"Whey Protein,Proteína,2,kg,150.00,200.00,50.00,100.00,33.33"
"Creatina,Creatina,1,kg,80.00,100.00,20.00,20.00,25.00"
```

Totalizadores exibidos:

- Soma total de compras
- Soma total de vendas potenciais
- Lucro total potencial
- Margem média de lucro

- Quantidade total de itens
- Número de categorias

---

src/services/ - Serviços Externos

firebase.js - Configuração Firebase

```
Localização: src/services/firebase.js
Função: Configuração central dos serviços Firebase
```

Configuração do projeto:

```
const firebaseConfig = {
  apiKey: "AIzaSyD...",                    // Chave da API
  authDomain: "inventario-8b388.firebaseapp.com",
  projectId: "inventario-8b388",           // ID do projeto
  storageBucket: "inventario-8b388.appspot.com",
  messagingSenderId: "123456789",
  appId: "1:123456789:web:abcd1234..."     // ID da aplicação
};
```

Serviços inicializados:

```
import { initializeApp } from 'firebase/app';
import { getAuth, GoogleAuthProvider } from 'firebase/auth';
import { getFirestore } from 'firebase/firestore';

const app = initializeApp(firebaseConfig);

export const auth = getAuth(app);           // Autenticação
export const db = getFirestore(app);       // Banco de dados
export const googleProvider = new GoogleAuthProvider(); // Provedor Google
```

Configurações de segurança:

- API Key restrita por domínio
- Firestore Rules protegendo dados por usuário
- OAuth configurado apenas para domínios autorizados

---

src/App.js - Componente Principal

```
Localização: src/App.js
Função: Estrutura principal e roteamento da aplicação
```

Estrutura do componente:

```
function App() {
  return (
    <AuthProvider>                    {/* Contexto de autenticação global */}
      <Router>                        {/* Roteamento da aplicação */}
        <Routes>
          <Route path="/login" element={<Login />} />

          <Route path="/*" element={
            <ProtectedRoute>          {/* Proteção de rotas privadas */}
              <InventoryProvider>     {/* Contexto do inventário */}
                <div className="app">
                  <Header />          {/* Barra de navegação */}
                  <main>
                    <Routes>
                      <Route path="/" element={<Dashboard />} />
                      <Route path="/inventory" element={<Inventory />} />
                      <Route path="/reports" element={<Reports />} />
                    </Routes>
                  </main>
                </div>
              </InventoryProvider>
            </ProtectedRoute>
          } />
        </Routes>
      </Router>
    </AuthProvider>
  );
}
```

Hierarquia de contextos:

1. **AuthProvider** (global)
    - Gerencia estado de autenticação
    - Disponível em toda aplicação
2. **Router** (navegação)
    - Controla roteamento entre páginas
    - Histórico de navegação
3. **ProtectedRoute** (segurança)
    - Verifica autenticação
    - Redireciona se necessário
4. **InventoryProvider** (dados)

- Gerencia dados do inventário
- Disponível apenas para usuários logados

`src/index.js` - Ponto de Entrada

```
Localização: src/index.js
Função: Inicialização e renderização da aplicação
```

Responsabilidades:

- Importar dependências globais (React, CSS, polyfills)
- Configurar tema do Material-UI
- Renderizar aplicação no DOM
- Configurar StrictMode para desenvolvimento

ARQUIVOS DE CONFIGURAÇÃO

`package.json` - Configuração do Projeto

```
{
  "name": "inventario-suplementos",
  "version": "1.0.0",
  "private": true,
  "dependencies": {
    "react": "^19.1.0",
    "react-dom": "^19.1.0",
    "react-router-dom": "^7.7.0",
    "react-hook-form": "^7.60.0",
    "@mui/material": "^7.2.0",
    "@mui/x-data-grid": "^7.2.0",
    "firebase": "^11.1.0",
    "axios": "^1.10.0"
  },
  "scripts": {
    "start": "react-scripts start",      // Desenvolvimento
    "build": "react-scripts build",      // Build de produção
    "test": "react-scripts test",        // Testes
    "eject": "react-scripts eject",      // Ejetar CRA
    "deploy": "firebase deploy --only hosting"  // Deploy Firebase
  }
}
```

`firebase.json` - Configuração Firebase

```
{
  "hosting": {
    "public": "build",                    // Pasta de build
    "ignore": ["firebase.json", "**/.*", "**/node_modules/**"],
    "rewrites": [
      {
        "source": "**",                   // Todas as rotas
        "destination": "/index.html"      // Redirect para SPA
      }
    ],
    "headers": [
      {
        "source": "**/*.@(js|css)",      // Cache para assets
        "headers": [
          {
            "key": "Cache-Control",
            "value": "max-age=31536000"
          }
        ]
      }
    ]
  }
}
```

.gitignore - Arquivos Ignorados

```
node_modules/        # Dependências
build/              # Build de produção
.env                # Variáveis de ambiente
.firebase/          # Cache do Firebase
.vscode/           # Configurações do VS Code
```

FLUXO DE DADOS COMPLETO

1. Inicialização da Aplicação

```
index.js → App.js → AuthProvider
   ↓
Verifica autenticação salva
   ↓
Se autenticado: Dashboard
Se não: Login
```

2. Processo de Login

```
Login.js → signInWithGoogle()
   ↓
Google OAuth Popup
   ↓
Firebase Auth Token
   ↓
AuthContext atualizado
   ↓
Redirect para Dashboard
```

3. Carregamento de Dados

```
Dashboard carregado
   ↓
InventoryProvider inicializado
   ↓
fetchSupplements() chamado
   ↓
Firestore query (where userId == currentUser.uid)
   ↓
Dados carregados no contexto
   ↓
Componentes re-renderizam
```

4. Operação CRUD (Exemplo: Adicionar)

```
Usuário preenche formulário
   ↓
SupplementForm → onSubmit
   ↓
Validação dos dados
   ↓
addSupplement() no contexto
   ↓
Firestore addDoc()
   ↓
Estado local atualizado
   ↓
UI re-renderizada
   ↓
Feedback para usuário
```

5. Cálculos Financeiros

```
Dados carregados
   ↓
```

```
getTotalProfit() chamado
   ↓
supplements.reduce() executado
   ↓
Soma: (precoVenda - precoCompra) * quantidade
   ↓
Resultado exibido no Dashboard
```

SEGURANÇA E PROTEÇÃO

Firestore Security Rules

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /supplements/{document} {
      // Usuário só acessa seus próprios dados
      allow read, write: if request.auth != null
        && request.auth.uid == resource.data.userId;

      // Apenas usuários autenticados podem criar
      allow create: if request.auth != null
        && request.auth.uid == request.resource.data.userId;
    }
  }
}
```

Proteção de Rotas

```
// ProtectedRoute.js
if (!user) {
  return <Navigate to="/login" replace />;
}
return children;
```

Validação de Dados

- Validação client-side (React Hook Form)
- Validação server-side (Firestore Rules)
- Sanitização de inputs
- Tipos de dados consistentes

RESPONSIVIDADE

Breakpoints Material-UI

```
// Configuração responsiva
{
  xs: 0,      // Extra small devices (phones)
  sm: 600,    // Small devices (tablets)
  md: 900,    // Medium devices (small laptops)
  lg: 1200,   // Large devices (desktops)
  xl: 1536    // Extra large devices (large desktops)
}
```

Layout Adaptativo

```
// Dashboard cards
<Grid container spacing={3}>
  <Grid item xs={12} sm={6} md={3}>
    <StatCard />
  </Grid>
</Grid>

// Sidebar mobile
<Drawer
  variant="temporary"          // Mobile
  sx={{ display: { xs: 'block', md: 'none' } }}
/>
<Drawer
  variant="permanent"          // Desktop
  sx={{ display: { xs: 'none', md: 'block' } }}
/>
```

🚀 DEPLOY E HOSPEDAGEM

Build de Produção

```
npm run build
```

Resultado:

- Código minificado
- Assets otimizados
- Service Worker para PWA
- Pasta `build/` criada

Deploy no Firebase

```
firebase deploy --only hosting
```

Processo:

1. Upload da pasta `build/`
2. Configuração de CDN
3. SSL automático
4. URL disponibilizada

URL de Produção

```
https://inventario-8b388.web.app
```

📈 MÉTRICAS E ANÁLISES

Indicadores Calculados

1. **Valor Total Investido**: Soma de precoCompra × quantidade
2. **Receita Potencial**: Soma de precoVenda × quantidade
3. **Lucro Bruto**: Receita - Investimento
4. **Margem de Lucro**: (Lucro ÷ Investimento) × 100
5. **ROI**: Retorno sobre investimento
6. **Ticket Médio**: Valor médio por produto

Alertas Automáticos

- 🟨 **Estoque baixo**: quantidade ≤ estoqueMinimo
- 🟧 **Vencimento**: dataVencimento ≤ hoje + 30 dias
- ⚠️ **Margem negativa**: precoVenda < precoCompra
- 📊 **Análise**: Comparações e tendências

🎯 BENEFÍCIOS DO SISTEMA

Para o Negócio

- ☑ **Controle total** do estoque em tempo real
- ☑ **Análise financeira** automatizada
- ☑ **Prevenção de perdas** por vencimento
- ☑ **Otimização de compras** baseada em dados
- ☑ **Relatórios** para tomada de decisão
- ☑ **Escalabilidade** com crescimento do negócio

Para o Usuário

- ☑ **Interface intuitiva** e responsiva
- ☑ **Acesso multiplataforma** (web/mobile)

- ☑ **Dados sincronizados** em tempo real
- ☑ **Alertas automáticos** importantes
- ☑ **Exportação** de relatórios
- ☑ **Segurança** dos dados pessoais

Técnicos

- ☑ **Arquitetura moderna** e escalável
- ☑ **Performance otimizada** com CDN
- ☑ **Backup automático** no Firebase
- ☑ **Updates automáticos** sem downtime
- ☑ **Monitoramento** de erros
- ☑ **Manutenibilidade** do código

🔧 TECNOLOGIAS DETALHADAS

React.js 19.1.0

- **Hooks utilizados**: useState, useEffect, useContext, useCallback
- **Patterns**: Context API, Custom Hooks, HOCs
- **Performance**: React.memo, useMemo, useCallback

Firebase

- **Auth**: Google OAuth 2.0, JWT tokens
- **Firestore**: NoSQL database, real-time sync
- **Hosting**: CDN global, SSL automático

Material-UI 7.2.0

- **Componentes**: 40+ componentes utilizados
- **Theming**: Tema customizado com cores da marca
- **Data Grid**: Tabelas avançadas com 20+ recursos

React Router DOM 7.7.0

- **Roteamento**: SPA com lazy loading
- **Proteção**: Rotas privadas e públicas
- **Navigation**: Programática e declarativa

Este documento representa a estrutura completa e detalhada do Sistema de Inventário de Suplementos, incluindo todos os aspectos técnicos, funcionais e arquiteturais do projeto.

Data de criação: Julho 2025
Versão: 1.0
Status: Produção ativo em https://inventario-8b388.web.app