🚀 Guia Completo de Deploy

📋 Pré-requisitos

- ✅ Flutter SDK 3.16+ instalado
- ✅ Conta Firebase ativa
- ✅ Conta MercadoPago (sandbox)
- ✅ Node.js 18+ instalado
- ✅ Git configurado

🔥 Configuração Firebase

1. Criar Projeto Firebase

```
# Acesse: https://console.firebase.google.com
# Clique em "Adicionar projeto"
# Siga as instruções de criação
```

2. Configurar Serviços

No Firebase Console, ative:
- ✅ **Authentication** (Email/Password + Google)
- ✅ **Firestore Database**
- ✅ **Cloud Functions**
- ✅ **Hosting**
- ✅ **Cloud Messaging**
- ✅ **Analytics**
- ✅ **Crashlytics**

3. Instalar CLI Tools

```
# Firebase CLI
npm install -g firebase-tools

# FlutterFire CLI
dart pub global activate flutterfire_cli

# Login no Firebase
firebase login
```

4. Configurar Projeto

```
# No diretório do projeto Flutter
flutterfire configure

# Selecione seu projeto Firebase
# Selecione plataformas: Web, Android (opcional), iOS (opcional)
```

5. Configurar Authentication

```
# No Firebase Console → Authentication → Sign-in method
# Ative "Email/password"
# Ative "Google" e configure:
#   - Nome do projeto
#   - Email de suporte
#   - Domínios autorizados
```

💳 Configuração MercadoPago

1. Criar Conta Developer

- Acesse: https://www.mercadopago.com.br/developers
- Crie uma aplicação
- Anote as credenciais de sandbox e produção

2. Configurar Credenciais

```
# Configurar no Firebase Functions
firebase functions:config:set \
  mercadopago.access_token="YOUR_ACCESS_TOKEN" \
  mercadopago.public_key="YOUR_PUBLIC_KEY"

# Para produção, use as chaves de produção
```

3. Configurar Webhooks

- URL do webhook: `https://YOUR_PROJECT.cloudfunctions.net/mercadopagoWebhook`
- Eventos: `payment`

🔧 Configuração do Projeto

1. Substituir Chaves

Edite os seguintes arquivos:

firebase_options.dart

```
// Substitua pelas suas chaves reais
apiKey: 'YOUR_ACTUAL_API_KEY',
appId: 'YOUR_ACTUAL_APP_ID',
// ... etc
```

web/index.html

```
<!-- Substitua pela sua chave pública do MercadoPago -->
window.mp = new MercadoPago('YOUR_MERCADOPAGO_PUBLIC_KEY');
```

services/notification_service.dart

```
// Substitua pela sua chave VAPID
vapidKey: 'YOUR_VAPID_KEY',
```

2. Configurar Firestore Rules

```
# Copie as regras do arquivo firestore.rules
firebase deploy --only firestore:rules
```

3. Deploy Cloud Functions

```
cd functions
npm install
npm run build
firebase deploy --only functions
```

📊 Popular Dados Iniciais

1. Executar Script

```
# No diretório do projeto
dart run scripts/populate_firestore.dart
```

2. Verificar Dados

- Acesse Firebase Console → Firestore
- Verifique se as coleções foram criadas:
- `categories`
- `restaurants`
- `products`
- `coupons`

🌐 Build e Deploy

1. Build para Produção

```
# Limpar build anterior
flutter clean

# Instalar dependências
flutter pub get

# Build para web
flutter build web --web-renderer html --release
```

2. Deploy no Firebase Hosting

```
# Deploy completo
firebase deploy

# Ou apenas hosting
firebase deploy --only hosting
```

3. Verificar Deploy

- Acesse a URL fornecida pelo Firebase
- Teste todas as funcionalidades principais
- Verifique se PWA pode ser instalada

🔄 CI/CD (Opcional)

1. Configurar GitHub Actions

- Copie arquivo `.github/workflows/deploy.yml`
- Configure secrets no GitHub:
- `FIREBASE_TOKEN`: Execute `firebase login:ci`

2. Auto-deploy

- Push para branch `main` = deploy automático
- Pull requests = testes automáticos

📱 Configuração PWA

1. Verificar Manifest

- `web/manifest.json` configurado
- Ícones PWA incluídos
- Service Worker ativo

2. Testar Instalação

- Abra o app no Chrome/Edge
- Clique no ícone de instalação na barra de endereços
- Teste funcionalidades offline

🔔 Configurar Notificações

1. Gerar Chave VAPID

```
# No Firebase Console → Cloud Messaging → Web configuration
# Clique em "Generate key pair"
# Copie a chave gerada
```

2. Atualizar Código

- Substitua `YOUR_VAPID_KEY` no código
- Deploy novamente

3. Testar Notificações

- Faça login no app
- Permita notificações no browser
- Teste envio via Firebase Console

🧪 Testes

1. Testes Unitários

```
flutter test
```

2. Testes E2E (Opcional)

```
# Instalar integration_test
flutter pub add integration_test

# Executar testes
flutter drive --driver=test_driver/integration_test.dart --target=integration_test/
app_test.dart
```

📈 Monitoramento

1. Analytics

- Eventos configurados automaticamente
- Acesse Firebase Console → Analytics

2. Crashlytics

- Crashes reportados automaticamente
- Acesse Firebase Console → Crashlytics

3. Performance

- Métricas automáticas
- Acesse Firebase Console → Performance

🔧 Troubleshooting

Problema: CORS Errors

```
# Configure CORS no firebase.json
"headers": [
  {
    "source": "**/*",
    "headers": [
      {
        "key": "Access-Control-Allow-Origin",
        "value": "*"
      }
    ]
  }
]
```

Problema: PWA não instala

- Verifique manifest.json
- Confirme service worker ativo
- Teste em HTTPS

Problema: Notificações não funcionam

- Verifique chave VAPID
- Confirme service worker
- Teste permissões

Problema: Pagamentos falham

- Verifique credenciais MercadoPago
- Confirme webhooks configurados
- Teste em sandbox primeiro

🎯 Checklist Final

Antes de colocar em produção:

- [ ] Todas as chaves configuradas
- [ ] Firestore rules aplicadas
- [ ] Cloud Functions deployadas
- [ ] Dados iniciais populados
- [ ] PWA instalável
- [ ] Notificações funcionando
- [ ] Pagamentos testados
- [ ] Analytics ativo
- [ ] Domínio personalizado (opcional)

🚀 Go Live!

```
# Build final
flutter build web --web-renderer html --release

# Deploy produção
firebase deploy

# Verificar URL
echo "App disponível em: https://YOUR_PROJECT.web.app"
```

🎉 **Parabéns! Seu app está no ar!**

📞 Suporte

Para dúvidas sobre:
- **Flutter**: https://docs.flutter.dev
- **Firebase**: https://firebase.google.com/docs
- **MercadoPago**: https://www.mercadopago.com.br/developers

Próximo: Solicite o desenvolvimento do **App Restaurante** em uma nova sessão!