# 開発およびデプロイメントガイド

## 目次



## システム要件

- Node.js（最新バージョン）
- Bun Runtime
- Git
- VS Code（推奨）

### 必要なVS Code拡張機能

- Tailwind CSS IntelliSense
- ES7+ React/Redux/React-Native snippets
- Biome Extension

## インストールとセットアップ

1. プロジェクトのクローン:

```
git clone [repository-url]
cd react-vite-with-unit-test
```

2. 依存関係のインストール:

```
bun install
```

3. 環境設定:
   - `.env.example` を `.env.development` にコピー
   - 必要な環境変数を更新
4. Gitフックのセットアップ:

```
bun prepare
```

## 開発プロセス

### 1. 開発環境の起動

```
bun dev
```

### 2. コンポーネント開発

1. 新しいコンポーネントの作成:

```
bun generate
```

2. Storybookでの開発:

```
bun storybook
```

### 3. コーディングワークフロー

1. 新しいブランチの作成:

```
git checkout -b feature/機能名
```

2. コミット前のコードフォーマット:

```
bun format
bun lint
```

3. コードチェック:

```
bun check
```

## テスト

### 1. ユニットテスト

```
# 全テストの実行
bun test

# ウォッチモードでテスト実行
bun test:watch

# カバレッジの確認
bun test:coverage
```

### 2. Storybookテスト

```
# Storybookの起動
bun storybook

# Storybookのビルド
bun build-storybook
```

## デプロイメント

### 1. プロダクションビルド

```
bun build
```

### 2. ビルドのプレビュー

```
bun preview
```

### 3. Dockerデプロイメント

```
# Dockerイメージのビルド
docker build -t react-vite-app .

# コンテナの実行
docker run -p 8080:80 react-vite-app
```

## ルールと基準

### 1. コードスタイル

- Biomeを使用してフォーマットとリント
- TypeScriptのstrictモードに従う
- 関数コンポーネントとフックを使用

### 2. Gitコミット

- conventional commitsを使用
- 各コミットは全テストをパスする必要がある
- プレコミットフックでチェック:
  - コードフォーマット
  - リント
  - 型チェック
  - ユニットテスト

### 3. コンポーネント開発

1. ディレクトリ構造:

```
src/
  ├── components/
  │   ├── [ComponentName]/
  │   │   ├── index.tsx
  │   │   ├── [ComponentName].stories.tsx
  │   │   └── [ComponentName].test.tsx
  │   └── ...
  ├── hooks/
  ├── utils/
  └── ...
```

2. コンポーネントガイドライン:

- TypeScriptインターフェースでpropsを定義
- エラーバウンダリーの実装
- 必要に応じてReact.memoでパフォーマンス最適化
- すべてのケースのストーリーを作成

### 4. テストガイドライン

- 最小カバレッジ: 80%
- エッジケースのテスト
- 外部依存のモック
- testing-libraryのベストプラクティスに従う

### 5. パフォーマンス最適化

- ルートの遅延読み込み
- コード分割
- 画像最適化
- PWA実装

## CI/CDパイプライン

GitHub Actionsワークフローには以下が含まれます:

1. ビルドチェック

2. ユニットテスト
3. Storybookビルド
4. Dockerイメージビルド
5. ステージング/本番環境へのデプロイ

## モニタリングとロギング

1. エラーバウンダリーによるエラー追跡
2. パフォーマンスモニタリング
3. ユーザー分析

## セキュリティガイドライン

1. 依存関係のスキャン
2. 定期的なアップデート
3. セキュリティベストプラクティス
4. 環境変数の管理

## サポート

問題が発生した場合:

1. ドキュメントを確認
2. GitHubでイシューを作成
3. チームリーダーに連絡