Documentation Détailée de Benin Culture Hub

<https://github.com/Adebissi/tpfinaldclic>

Cette documentation fournit une explication complète de la structure de l'application, de ses fonctionnalités principales, et des choix de conception adoptés. L'application "Benin Culture Hub" est un projet Flutter conçu pour promouvoir la culture béninoise à travers des événements, une timeline historique, et une galerie d'art. Elle est cross-platform (Android et iOS) et utilise Firebase pour le stockage de données en temps réel.

Structure de l'Application

Le projet suit une architecture Flutter standard, avec une organisation modulaire pour faciliter la maintenance et l'extension. Voici l'arborescence principale du projet :

```

beninculturehub/

├── android/ Configuration Android (manifest, gradle, etc.)

├── ios/ Configuration iOS (Info.plist, Assets.xcassets, etc.)

├── lib/ Code source principal

│ ├── main.dart Point d'entrée, initialise Firebase et lance l'app

│ │ ├── page\_accueil.dart Écran d'accueil avec banner et teasers

│ │ ├── events\_screen.dart Écran des événements avec liste et recherche

│ │ ├── event\_add\_screen.dart Formulaire d'ajout d'événement

│ │ ├── event\_detail\_screen.dart Détails d'un événement

│ │ ├── history\_screen.dart Écran de l'histoire avec timeline

│ │ ├── history\_add\_screen.dart Formulaire d'ajout d'élément historique

│ │ ├── history\_detail\_screen.dart Détails d'un élément historique

│ │ ├── gallery\_screen.dart Écran de la galerie avec grille

│ │ ├── art\_piece\_add\_screen.dart Formulaire d'ajout d'œuvre d'art

│ │ └── art\_piece\_detail\_screen.dart Détails d'une œuvre d'art

│ ├── models/ Modèles de données

│ │ ├── event.dart Modèle pour les événements

│ │ ├── history\_item.dart Modèle pour les éléments historiques

│ │ └── art\_piece.dart Modèle pour les œuvres d'art

├── assets/ Ressources statiques

│ └── images/ Images comme default\_event.jpg et logo.png

├── build/ Fichiers générés (ignorés par Git)

├── pubspec.yaml Dépendances, assets, et configuration (ex. : flutter\_launcher\_icons)

├── README.md Instructions d'installation et d'utilisation

└── .gitignore Fichiers à ignorer (build, .idea, etc.)

```

- lib/main.dart : Initialise l'application, configure Firebase, et lance `PageAccueil` comme écran initial.

- screens/ : Chaque écran est un `StatefulWidget` ou `StatelessWidget` pour gérer l'UI et les interactions. Les écrans d'ajout et de détails sont séparés pour une modularité.

- models/ : Classes Dart pour les données (ex. : `Event`, `HistoryItem`, `ArtPiece`), avec méthodes `toMap` et `fromMap` pour la sérialisation Firestore.

- assets/ : Contient les images statiques comme l'icône par défaut (`default\_event.jpg`) et le logo (`logo.png`).

- Plateformes spécifiques : `android/` et `ios/` contiennent les configurations pour les permissions, icônes, et builds (ex. : AndroidManifest.xml, Info.plist).

Cette structure suit le principe de séparation des préoccupations : UI séparée des données, et modularité pour une scalabilité facile.

Fonctionnalités

L'application est centrée sur la promotion culturelle du Bénin. Voici une liste détaillée des fonctionnalités, organisées par section.

1. Navigation Globale

- BottomNavigationBar : Barre de navigation en bas avec 4 onglets (Accueil, Événements, Histoire, Galerie). Chaque onglet utilise `Navigator.pushReplacement` pour une transition fluide sans empilement de routes.

- Choix : Permet une navigation intuitive et persistante sur tous les écrans.

2. Accueil (`PageAccueil`)

- Banner hero avec un événement en vedette (placeholder pour une image et texte).

- Grid 2x1 de teasers (Histoire du Dahomey, Bronzes du Bénin) avec placeholders pour images.

- Recherche via icône dans l'AppBar (à implémenter).

- Fonctionnalité : Vue d'ensemble pour attirer l'utilisateur vers les sections principales.

3. Événements (`EventsScreen`)

- Liste dynamique d'événements depuis Firestore, triée par date.

- Recherche par titre/description et filtrage par localisation.

- Ajout d'événement via formulaire (`EventAddScreen`) avec caméra pour photo.

- Détails (`EventDetailScreen`) avec image Hero, description, favoris, partage, et bouton "Ajouter au calendrier" (à implémenter).

- Favoris : Mise à jour en temps réel dans Firestore.

- Fonctionnalité : Promotion des événements culturels comme le Festival Vodun.

4. Histoire (`HistoryScreen`)

- Timeline horizontale interactive avec éléments historiques depuis Firestore, triés par période.

- Ajout d'élément historique via formulaire (`HistoryAddScreen`) avec caméra.

- Détails (`HistoryDetailScreen`) avec image Hero, période, description, favoris, et partage.

- Section Quiz avec bouton "Tester vos connaissances" (à implémenter avec questions sur l'histoire béninoise).

- Fonctionnalité : Éducation sur l'histoire (Royaume du Dahomey, traite des esclaves).

5. Galerie (`GalleryScreen`)

- Grille 2 colonnes d'œuvres d'art depuis Firestore.

- Ajout d'œuvre via formulaire (`ArtPieceAddScreen`) avec caméra.

- Détails (`ArtPieceDetailScreen`) avec image Hero zoomable (via `photo\_view`), titre, artiste, description, favoris, et partage.

- Fonctionnalité : Présentation des arts traditionnels (Bronzes du Bénin, masques).

6. Fonctionnalités Transversales

- Caméra : Intégrée pour prendre des photos (sauvegardées localement, chemin stocké dans Firestore).

- Favoris : Basculés via icône cœur, mis à jour dans Firestore.

- Partage : Bouton pour partager (à implémenter avec `share\_plus`).

- Images par défaut : `default\_event.jpg` utilisé si aucune image n'est disponible.

- Animations : Hero pour transitions fluides entre listes et détails.

- Validation : Formulaires avec validation et messages d'erreur en rouge.

- Offline : Favoris stockés localement via `sqflite` (à implémenter pour une V2).

Choix de Conception

1. Technologies

- Flutter : Choisi pour le développement cross-platform rapide, avec une UI native et une communauté active. Alternative : React Native, mais Flutter offre une meilleure performance pour les animations.

- Firebase Firestore : Backend sans serveur pour un stockage en temps réel (événements, histoire, galerie). Choix : Scalable et facile à intégrer, sans gestion d'infrastructure. Alternative : Supabase pour du SQL, mais Firestore convient pour des données non relationnelles.

- Camera Plugin : Pour la prise de photos, avec sauvegarde locale pour éviter les coûts de stockage cloud. Alternative : Firebase Storage pour un partage multi-appareils, mais limité pour simplifier.

- PhotoView : Pour le zoom interactif dans la galerie, respectant le wireframe.

2. UI/UX

- Palette de couleurs : Vert (228B22) pour la nature et l'héritage béninois, orange (FF4500) pour l'énergie culturelle, fond gris clair (F5F5F5) pour une lecture confortable. Minimalisme pour éviter le clutter.

- Police : Roboto pour la lisibilité et le contraste élevé (16-24pt), conforme aux guidelines Android/iOS.

- Layout : SingleChildScrollView pour une navigation fluide sur tous les écrans. Grilles et listes pour une organisation claire.

- Animations : Hero pour transitions immersives (images), FadeTransition pour chargements (ex. : opacity sur images).

- Accessibilité : Contraste élevé, tailles de texte ajustables, support screen readers via Flutter.

3. Architecture

- MVC : Models pour les données, Views pour l'UI, Controllers pour la logique (ex. : gestion d'état avec Provider pour les favoris).

- Modularité : Écrans et modèles séparés pour une extension facile (ex. : ajouter un quiz).

- Écoconception : Chargement lazy des images, minimisation des requêtes Firestore pour réduire la consommation d'énergie.

- Limites : Images locales non synchronisées entre appareils (choix pour simplifier, améliorable avec Storage).

- Sécurité : Règles Firestore pour limiter les accès (ex. : authentification optionnelle pour contributions).

Améliorations Futures

- Intégrer Firebase Storage pour les images partagées.

- Implémenter le quiz avec questions dynamiques.

- Ajouter authentification Firebase pour utilisateurs.

- Tests unitaires avec `flutter\_test`.

- Localisation (français/anglais) avec `intl`.

Pour toute contribution, ouvre une PR sur le dépôt Git.