Twins Software Factory

DOSSIER de specification technique

Application Mobile DIBNEWS

# Suivi du document

|  |  |  |  |
| --- | --- | --- | --- |
| Mise à jour | | | |
| Version | **Date** | **Auteurs** | **Objet** |
| 1.0 | 22/03/2021 | HFI | Création du document |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Cycle de validation

|  |  |  |
| --- | --- | --- |
| Validation | | |
| Version | **Nom** | **Date / Visa** |
| 1.0 | Hamza EL FOUNASSI | 22 Mars 2021 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Table des matières

[Suivi du document 1](#_Toc76054581)

[Cycle de validation 1](#_Toc76054582)

[Introduction 3](#_Toc76054583)

[Objet du document 3](#_Toc76054584)

[Domaine d’application 3](#_Toc76054585)

[Description générale 3](#_Toc76054586)

[Gestion de la sécurité 3](#_Toc76054587)

[Technologies utilisées 4](#_Toc76054588)

[Structure du projet 4](#_Toc76054589)

[Charte graphique 5](#_Toc76054590)

[Couleurs 5](#_Toc76054591)

[Logo 5](#_Toc76054592)

[L’aspect général 7](#_Toc76054593)

[Écran d'actualité 7](#_Toc76054594)

[Le menu natif principal 8](#_Toc76054595)

[Ecran à propos 9](#_Toc76054596)

[L'écran d'options 10](#_Toc76054597)

[Modèle des données 12](#_Toc76054598)

[Données locales 12](#_Toc76054599)

[Données API 12](#_Toc76054600)

[Objet Catégorie 13](#_Toc76054601)

[Objet article 13](#_Toc76054602)

[Description des requêtes d’API 14](#_Toc76054603)

[/wp/v2/categories 14](#_Toc76054604)

[/wp/v2/posts 14](#_Toc76054605)

[Plus d'informations 14](#_Toc76054606)

[Initialisation de l'environnement de développement et autres documentations 14](#_Toc76054607)

[Les exigences : 15](#_Toc76054608)

[Démarrage du projet 15](#_Toc76054609)

[Documentations utiles : 16](#_Toc76054610)

# Introduction

## Objet du document

L’objet de ce document est de définir les spécifications techniques détaillées de

L’application DIB NEWS.

Les spécifications techniques détaillées ont pour but de décrire précisément :

L’ensemble des fonctionnalités de l’application.

Les objets manipulés, leurs buts et leurs principes de fonctionnement.

Les écrans utilisateurs mettant en œuvre les fonctionnalités de l’application.

Toutes les fonctionnalités prévues lors de la phase de conception sont précisées dans ce document en indiquant l’implémentation de ces fonctionnalités dans l’application.

## Domaine d’application

Ce dossier de spécifications technique détaillées est applicable pendant la phase de développement de l’application DIB News.

Le fonctionnement de l’application sera conforme aux éléments présents dans ce dossier.

# Description générale

L'application se connecte au site Web principal dibnews.ma à l'aide de l'API RESTful native WordPress, puis affiche les articles dans une mise en page diaporama.

L'utilisateur peut glisser entre les diaporamas ou appuyer sur l'écran pour visionner la vidéo/les détails de l'article.

Les détails de l'article sont affichés sous une forme modale, afin que l'utilisateur puisse rapidement afficher les détails et revenir au même écran qu'il voyait auparavant.

# Gestion de la sécurité

Toutes les requêtes HTTP effectuées par l'application sont cryptées avec le certificat SSL inclus dans le nom de domaine du site Web principal.

L'application ne consomme aucune autre API ou interface distante autre que le site Web principal.

# Technologies utilisées

L’application mobile est codée en Javascript ECMA-262 et utilise le Framework React-Native.

React-native fournit tous les éléments nécessaires à la réalisation de notre application, et apporte plusieurs éléments essentiels :

* La portabilité : un code React-native compile indifféremment sous les deux systèmes d'exploitation ciblés, à savoir Android et iOS.
* Soutenu par Facebook, React-native est utilisé dans des projets professionnels de grande envergure (tels Microsoft apps). Il est en développement perpétuel et possède une communauté active. Cela nous assure la viabilité à long terme de ce Framework.

React-native est basé sur node js et son gestionnaire de dépendances (NPM ou YARN)

Plus de technologies et d'outils utilisés pour codée l'application et ils sont comme suit :

* redux: pour la synchronisation de l’état de l’application
* vector-icons : pour les icône vector (SVG)
* expo-av : lecteur video native
* i18n-js : pour l’internationalisation de l’application
* i18next : pour l’internationalisation
* moment : outil des dates

# Structure du projet

La structure du projet a été simplifiée à comprendre et se présente comme suit:

* **assets (dossier)** :dossier des ressources (images, video .. etc)
* **commons (dossier)**  : données de langue (non utilisées), relatives au module d'internationalisation
* **Components** **(dossier)** : contient tous les composants nécessaires utilisés dans l'interface utilisateur de l'application, tels que le composant d'en-tête, le composant de bouton, etc.
* **Data** **(dossier)** : pour les modèles
* **Global** **(dossier)** : contient des fichiers de feuille de style globaux et des outils helpers.
* **Languages** **(dossier)** : contient les fichiers de langues utilisées
* **Nav** **(dossier)** : composants de navigation
* **Redux** **(dossier)** : contient la structure de fichier de base "redux"
* **Screens** **(dossier)** : contient toutes les interfaces de l'application, plus un ensemble de propositions utilisées dans la version de maquette
* **Storage (dossier)**: module Twins de stockage
* **App.js (fichier)** : le script principal de l'application
* **app.json** **(fichier)**: Descripteur du projet
* **package.json** **(fichier)**: Descripteur de dépendances (format node.js)
* **i18n.js** **(fichier)**: script du module d’internationalisation

# Charte graphique

## Couleurs

Les couleurs utilisées et leurs codes hexadécimaux :

Background : PRIMARY #343434

Primary: PRIMARY #c0c0c0

Secondary: PRIMARY #ffffff (blanc)

Accent 1 : #ee312e #ee312e

Accent 2 : #63bc46 #63bc46

Accent 3 : #00a6e5 #00a6e5

## Logo

|  |  |  |
| --- | --- | --- |
| Logo principal | Icône launcher Android | Icône iOS |
|  |  |  |

# L’aspect général

## Écran d'actualité



Même mise en page utilisée à la fois dans l'écran d'accueil et dans les écrans des catégories

L'en-tête d'écran se compose du même en-tête partagé sur tous les écrans,

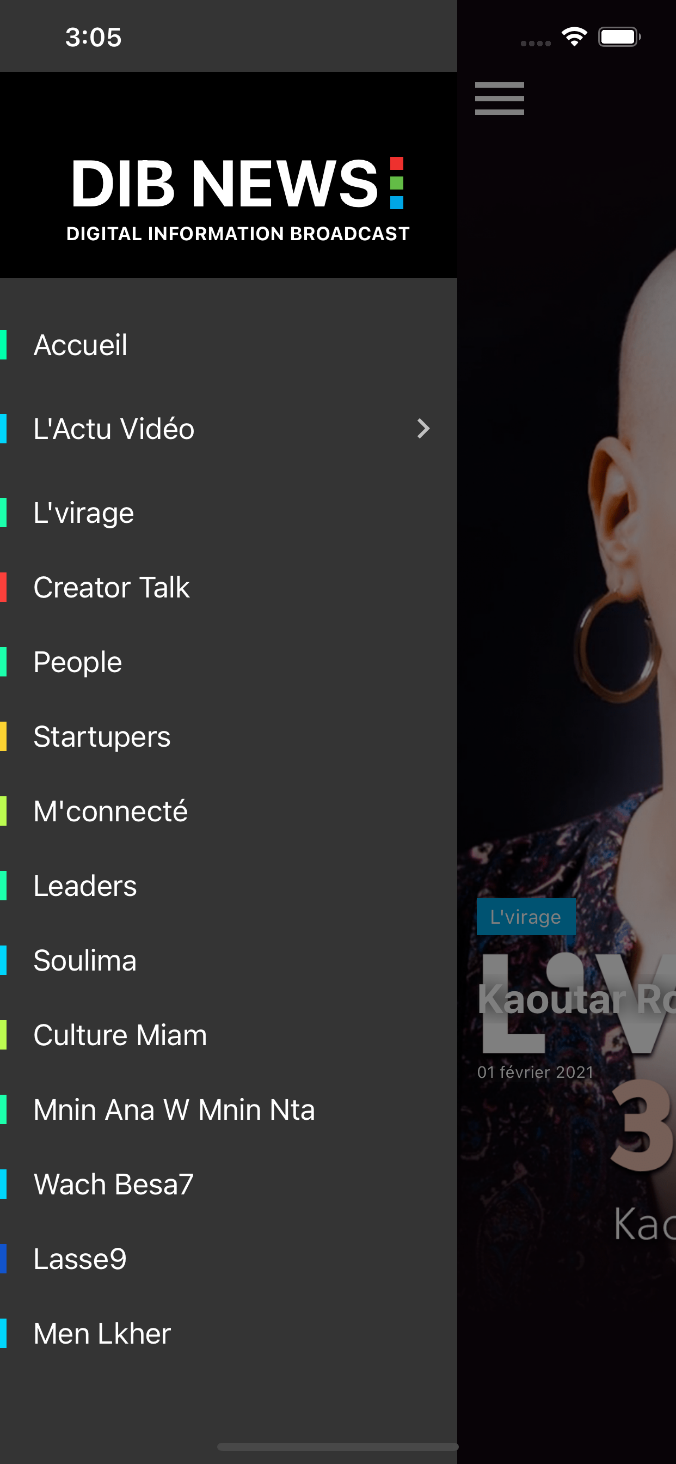
L'en-tête contient :

1 - à gauche un bouton pour déclencher la fonction qui gère la visibilité du menu natif.

2 - à droite : Le logo principal de l'application, et il peut déclencher plusieurs actions en fonction du contexte et de l'état de l'application :

* Peut amener l'utilisateur à la première diapositive lorsqu'il est sur l'écran d'accueil.
* Peut revenir à l'écran d'accueil lorsque l'utilisateur est sur un autre écran.
* Peut fonctionner comme un bouton "retour" dans un certain contexte

## Le menu natif principal

Le menu natif dynamique est généré pendant la période de démarrage de l'application, les éléments de menu sont séparés en deux catégories :

- Les éléments codés :

- Écran d'accueil

- À propos de l'application

- Partager avec des amis

- Les liens réseaux sociaux

- Les éléments dynamiques :

* Toutes les catégories du site, en respectant la hiérarchie définie.

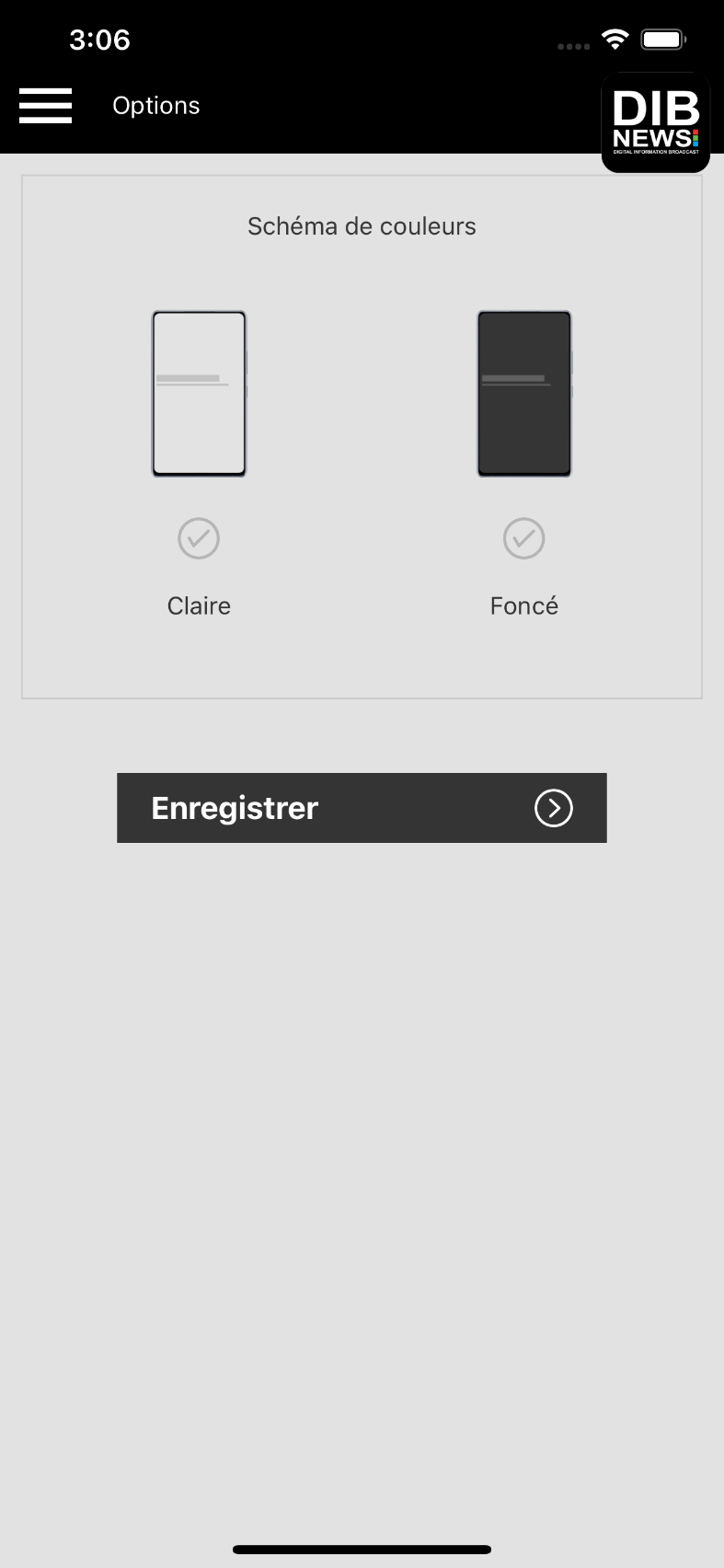
## Ecran à propos

Il s'agit d'un écran basé sur un contenu textuel donnant des informations sur l'application et des liens vers les différents comptes réseaux sociaux de DIB news.

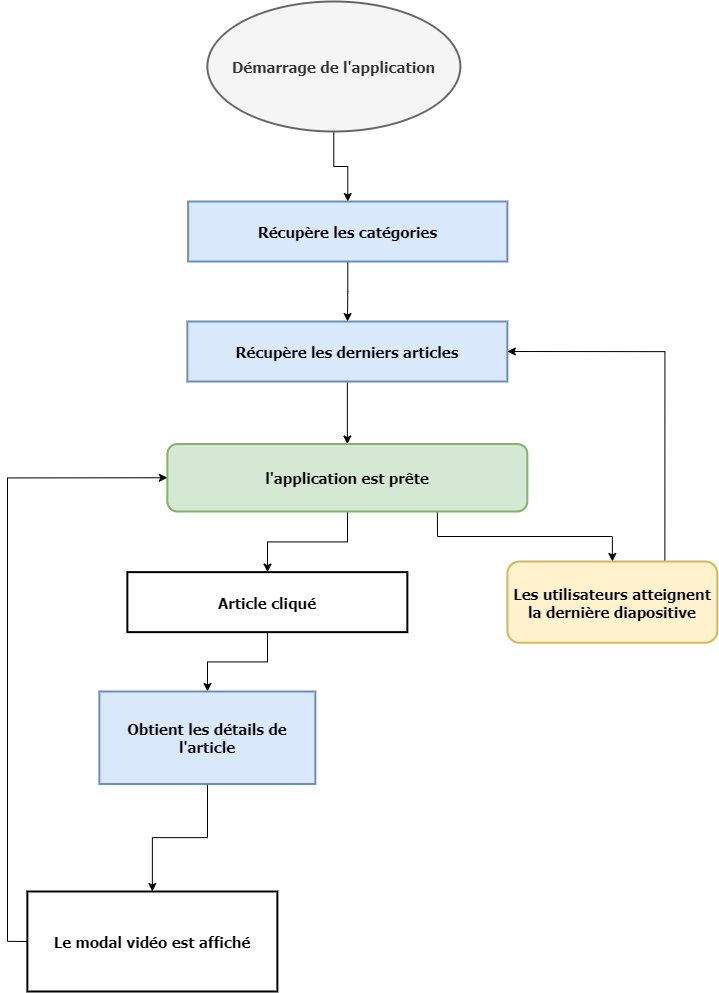


## L'écran d'options

L’écran d’options fournit pour le moment une option et c’est la possibilité de basculer entre les deux schémas de couleurs différents inclus dans l’application, le thème sombre et clair.



Cycle de vie de l’application



# Modèle des données

## Données locales

Référence du code source :

|  |  |  |
| --- | --- | --- |
| Module | Chemin | Méthodes |
| Storage | /storage/TwStorage.js | *setStorage*  *getStorage*  *getStorages* |

L'application utilise le stockage local natif pour stocker les préférences de l'utilisateur et les données temporaires extraites de l'API distante.

Les clés stockées :

* Clé « \_\_data » :
  + Language : Langue utilisateur actuelle
  + Dark : valeur 1 si le thème est sombre
  + is\_first\_time : valeur 1 si c'est la première exécution
  + initialized : date de la dernière initialisation
* Clé « cats » :
  + Tableau d'objets contenant toutes les catégories disponibles

## Données API

Référence du code source :

|  |  |  |
| --- | --- | --- |
| Module | Chemin | Méthodes |
| API | /storage/TwAPIClient.js | *POST*  *Query* |
| NewsData | /data/NewsData.js | *fetchNewsData*  *testGetPosts*  *fetchNewsByCatData*  *fetchCategories* |

Le module "NewsData" utilise le module "API" qui utilise la méthode native "Fetch" pour récupérer les données de l'API et les analyser.

### Objet Catégorie

|  |  |  |
| --- | --- | --- |
| Propriété | Type | Détail |
| Id | Int | Identifiant |
| count | Int | Nombre des articles |
| Description | Text | description |
| Link | Text | lien vers la catégorie |
| name | text | Nom |
| slug | text | nom court (slug) |
| taxonomy | text | Type |
| parent | int | Identifiant de la catégorie parente |
| meta | array | Données méta |

### Objet article

|  |  |  |
| --- | --- | --- |
| Propriété | Type | Détail |
| Id | Int | Identifiant |
| Title | Objet |  |
| Content | Objet |  |
| Excerpt | Objet |  |
| Date | Json datetime | Nombre des articles |
| date\_gmt |
| modified |
| modified\_gmt |
| Status | Text | Statut de l’article |
| Link | Text | Lien vers la catégorie |
| name | text | Nom |
| slug | text | Nom court (slug) |
| Type | text | Type |
| parent | int | Identifiant de la catégorie parente |
| categories | array |  |
| featured\_media | int |  |
| category\_name | Text |  |
| featured\_image\_src | Text |  |
| featured\_video\_src | text |  |

# Description des requêtes d’API

L'application utilise l'API restful wordpress native, et nous utilisons les routes de ressources suivantes :

## /wp/v2/categories

Cette route répond avec un tableau JSON contenant toutes les catégories disponibles.

L'application fait cette requête une seule fois à chaque exécution.

## /wp/v2/posts

Cette route est utilisée une fois lorsque l'application démarre pour préparer des articles d'actualité pour l'écran principal, et une fois à chaque fois que l'utilisateur atteint la fin des articles chargés.

Nous utilisons la même route pour récupérer les articles pour la vue des catégories.

## Plus d'informations

Pour plus d'informations sur ces ressources, veuillez consulter la documentation officielle de l'API RESTful de WordPress sur le lien suivant :

<https://developer.wordpress.org/rest-api/>

# Initialisation de l'environnement de développement et autres documentations

L'application a été développée en utilisant le langage et les Framework précédemment cités, dans cette section, nous allons lister toute la documentation utile et les étapes pratiques pour l'initiation de l'environnement de développement.

## Les exigences :

* Node js (<https://nodejs.org/>)
* NPM (https://www.npmjs.com/)
* React-native (<https://reactnative.dev>)
* Pour Android sous Windows : Emulateur Android (Nous recommandons le AVD de l’SDK Android inclus sur Android Studio)
* Pour iOS sous MAC : XCode

## Démarrage du projet

Après avoir installé tous les outils nécessaires, veuillez procéder à l'installation de toutes les dépendances à l'aide de la commande suivante :

* **npm install**

Cette commande effectuera toutes les actions nécessaires pour installer et configurer les dépendances du projet, et cela peut prendre un certain temps en fonction de la machine de travail.

Une fois que tout est installé, vous pouvez démarrer le projet sur un émulateur ou un périphérique physique en exécutant la commande suivante pour Android sur Windows

* **react-native run-android**

Ou la commande suivante pour iOS sur MAC :

* **react-native run-ios**

## Documentations utiles :

WordPress native API: <https://developer.wordpress.org/rest-api/>

React native: <https://reactnative.dev/docs/getting-started>

Redux: <https://redux.js.org/introduction/getting-started>

Fetch: <https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch>