Rapport de Projet - Page App.js

Conception : Utilisation de React Navigation pour la navigation entre les pages. Organisation des fonctionnalités principales avec des onglets en bas de l'écran.

Défis : Personnalisation des styles des onglets. Intégration des différentes pages et gestion de la navigation entre elles.

Solutions : Création d'une fonction CustomTabIcon pour personnaliser les icônes des onglets. Utilisation de piles de navigation pour gérer la navigation entre les pages. Personnalisation des styles pour correspondre au thème de l'application.

Rapport de Projet - Page HomePage.js

Conception : Utilisation d'une image de fond pour créer une atmosphère immersive. Utilisation de styles pour rendre le texte lisible sur l'image de fond. Structuration du contenu en sections pour une meilleure lisibilité.

Gestion des Couleurs : Choix de couleurs contrastées pour assurer une bonne visibilité du texte sur l'image de fond. Utilisation de la couleur de fond légèrement transparente pour améliorer la lisibilité du texte tout en préservant l'esthétique globale de l'application.

Défis : Intégration du texte sur l'image de fond tout en assurant la lisibilité. Création de styles attrayants qui s'harmonisent avec l'esthétique générale de l'application.

Solutions : Utilisation d'une couleur de fond légèrement transparente pour améliorer la lisibilité du texte. Choix de couleurs de texte contrastées pour assurer une bonne visibilité. Organisation du contenu en sections avec des titres distincts pour une navigation aisée.

Rapport de Projet - Page Personnages.js

Conception : Utilisation d'une image de fond pour créer une ambiance immersive. Utilisation de composants React Native tels que FlatList, TouchableOpacity pour afficher les personnages et permettre la navigation vers les détails des personnages.

Gestion des Couleurs : Choix de couleurs contrastées pour assurer une bonne visibilité du texte sur l'image de fond. Utilisation de gradients linéaires pour les bordures des éléments en fonction de la rareté des personnages, ajoutant une dimension visuelle intéressante.

Requêtes API : Utilisation de l'API pour récupérer la liste des personnages à afficher dans la page. Les données sont récupérées lors du montage du composant grâce à useEffect, puis affichées dans la FlatList.

Défis : Intégration des couleurs dynamiques en fonction de la rareté des personnages pour les bordures. Gestion de la mise en page pour assurer une présentation harmonieuse des personnages sur l'image de fond.

Solutions : Mise en place d'une fonction getGradientColorByRarity pour déterminer les couleurs en fonction de la rareté des personnages. Utilisation de styles dynamiques pour appliquer les couleurs correctes aux éléments en fonction des données des personnages. Utilisation de TouchableOpacity pour rendre les personnages cliquables et navigables vers leurs détails respectifs.

Rapport de Projet - Page CharacterDetails.js

Conception : Utilisation d'une ScrollView pour permettre le défilement du contenu. Utilisation d'une ImageBackground pour afficher l'image du personnage en arrière-plan. Utilisation de composants Text et View pour afficher les détails du personnage.

Gestion des Requêtes API : Utilisation de useEffect pour récupérer les détails du personnage à partir de l'API en fonction de l'ID du personnage passé en paramètre. Affichage d'un indicateur de chargement en attendant que les données du personnage soient récupérées.

Gestion des Couleurs : Utilisation de couleurs contrastées pour assurer une bonne lisibilité du texte sur l'image de fond. Utilisation de couleurs semi-transparentes pour les boîtes d'informations afin d'ajouter une dimension visuelle intéressante sans obstruer l'image de fond.

Défis : Intégration des données du personnage récupérées de l'API dans la mise en page. Gestion de l'aspect visuel pour assurer une présentation esthétique et lisible des détails du personnage sur l'image de fond.

Solutions : Utilisation de styles dynamiques pour afficher les détails du personnage tels que le nom, les points de vie, les points d'attaque, etc. Utilisation de différentes boîtes d'informations pour

organiser les détails du personnage de manière claire et visuellement attrayante. Utilisation de styles spécifiques pour assurer une bonne visibilité du texte sur l'image de fond.

Rapport de Projet - Page About.js

Conception : Utilisation de composants Text et TextInput pour afficher les informations sur l'équipe et le formulaire de contact. Utilisation de TouchableWithoutFeedback pour permettre la fermeture du clavier lorsqu'un utilisateur clique en dehors des champs de saisie.

Gestion des Couleurs : Utilisation de couleurs contrastées pour assurer une bonne lisibilité du texte sur le fond. Utilisation d'une couleur de fond semi-transparente pour le conteneur principal afin d'ajouter une légère opacité et de maintenir une lisibilité claire du texte.

Validation du Formulaire : Vérification des champs email, sujet et message avant l'envoi du formulaire. Affichage d'une alerte en cas de champ manquant.

Défis : Gestion de la fermeture du clavier lorsque l'utilisateur clique en dehors des champs de saisie. S'assurer que le formulaire de contact est facile à remplir et que les informations sont clairement présentées.

Solutions : Utilisation de TouchableWithoutFeedback pour détecter les clics en dehors des champs de saisie et fermer automatiquement le clavier. Utilisation de TextInput pour créer des champs de saisie interactifs avec une interface utilisateur familière. Utilisation de styles clairs et d'une mise en page bien structurée pour rendre le formulaire de contact convivial et accessible. Utilisation d'alertes pour fournir des retours d'information clairs à l'utilisateur en cas de problème avec le formulaire.

Rapport de Projet - Page More.js

Conception : La page More affiche des informations sur l'entreprise et propose un formulaire de contact. Elle utilise des composants Text, ScrollView, et TouchableOpacity pour afficher le contenu et permettre à l'utilisateur d'interagir avec le bouton de contact.

Gestion des Couleurs : Utilisation de couleurs contrastées pour assurer une bonne lisibilité du texte sur le fond. Le texte est affiché en noir ou en blanc selon le fond pour assurer un bon contraste visuel.

Requêtes API : Utilisation de fetch pour récupérer une image aléatoire à partir de l'API. L'image est utilisée comme arrière-plan de la page.

Interaction Utilisateur : Utilisation de Linking pour ouvrir l'application de messagerie par défaut lors du clic sur le bouton "Nous contacter".

Défis : Assurer une expérience utilisateur fluide et intuitive tout en intégrant des fonctionnalités telles que le chargement dynamique de l'image d'arrière-plan et l'ouverture de l'application de messagerie.

Solutions : Utilisation de React Hooks (useState et useEffect) pour gérer le chargement de l'image d'arrière-plan de manière asynchrone. Utilisation de TouchableOpacity pour créer un bouton interactif et d'un handler pour ouvrir l'application de messagerie par défaut. Utilisation de ScrollView pour permettre le défilement du contenu lorsque celui-ci dépasse l'écran.