

3PROJ - Sujet et Rendu

Ouvert le : lundi 18 décembre 2023, 00:00

À remettre : dimanche 19 mai 2024, 23:59

Les Bons Comptes Font Les Bons Amis

1 - Contexte du projet

La société "Les Bons Comptes Font Les Bons Amis" souhaite développer une application permettant à un groupe de personnes d'équilibrer des dépenses communes. Elle a pour cela lancé un appel d'offres pour confier ce projet à un sous-traitant.

Votre équipe est ainsi en compétition pour remporter ce contrat.

Vous devrez développer une application web et une application mobile afin de satisfaire les deux types de publics.

La charte graphique est également à définir.

2 - Description du projet

2.1 - Généralités

Le but principal de cette application est donc de permettre à un ensemble de personnes de gérer des dépenses communes.

Chaque membre du groupe peut ainsi déclarer une dépense en indiquant quels sont les autres membres en ayant profité. Cela pourra être soit l'ensemble du groupe, soit certaines personnes en particulier.

L'application déterminera alors les remboursements à effectuer pour équilibrer les comptes, c'est-à-dire "qui doit combien à qui ?".

2.2 - Fonctionnalités de l'application à implémenter

2.2.1 - Connexion

Un utilisateur pourra se connecter à l'application via un compte créé spécifiquement ou en utilisant un compte Facebook ou Google.

2.2.2 - Création d'un groupe

Tout utilisateur peut créer un groupe et inviter d'autres membres à le rejoindre. Une page dédiée à ce groupe sera alors créée.

Elle comportera un onglet pour les dépenses, un onglet indiquant le solde de chacun, un onglet pour les remboursements et une zone de messagerie instantanée.

2.2.3 - Création d'une dépense

Tout membre d'un groupe peut déclarer une dépense. Celle-ci sera constituée des éléments suivants :

- Un intitulé
- Un montant
- Une date
- Le membre ayant réglé cette dépense
- Un justificatif au format pdf ou image
- La catégorie de cette dépense (alimentation, frais de déplacement, activité, etc.)
- La liste des membres concernés par cette dépense et la pondération de chacun d'eux (par défaut tous les membres concernés auront le même poids)

Les justificatifs pourront être recherchés via un explorateur de fichiers puis uploadés ou directement scannés.

L'onglet dédié aux dépenses récapitulera la liste de celles-ci, et l'on pourra accéder au détail de chacune d'elles. Des fonctions de tri (par montant, membre, etc.) devront être implémentées.

2.2.4 - Soldes

Cet onglet proposera le solde de chacun des membres du groupe.

2.2.5 - Remboursements

Dans cet onglet il sera proposé une liste de remboursements afin d'équilibrer les comptes. Chacun de ces remboursements aura un expéditeur et un bénéficiaire. Il sera bien sûr pris en compte lors du calcul la liste des membres concernés par chacune des dépenses.

Cette liste de remboursements devra être optimisée, c'est-à-dire contenir le moins d'éléments possibles.

2.2.6 - Paiement

Un module de paiement via une application reconnue (Lydia, Lyf Pay, Paypal, etc.) devra être proposé.

Alternativement, un membre pourra uploader son RIB afin d'obtenir des remboursements par virements bancaires. Là aussi on indiquera manuellement les remboursements effectués une fois la transaction finie.

2.2.7 - Messagerie instantanée

Une zone de messagerie instantanée interne au groupe devra permettre d'effectuer des discussions de groupe mais aussi d'envoyer des messages privés. Les différents membres pourront ainsi demander et obtenir des explications sur les dépenses.

2.2.8 - Export

Les membres d'un groupe pourront exporter au format pdf et/ou au format csv un récapitulatif de l'ensemble des dépenses et/ou des remboursements.

2.2.9 - Statistiques

Chaque utilisateur pourra consulter ses statistiques de dépenses pour l'ensemble des groupes auxquels il a participé. Il pourra voir ses dépenses totales, par catégorie, par date, etc.

2.3 - Déploiement

2.3.1 - Architecture

Votre application doit comporter trois briques distinctes :

- un **serveur**, devant être une API REST ou GraphQL et devant implémenter l'ensemble des fonctionnalités précédemment énoncées,
- deux **clients** (**web** et **mobile** distincts) devant uniquement interagir avec le serveur.,
- une **base de données** (choix libre).

Aucune logique ne doit avoir lieu sur les clients qui ne servent que d'interfaces et redirigent les différentes requêtes vers le serveur.

2.3.2 - Containérisation

Le projet doit comporter un fichier **docker-compose.yml** à la racine du projet permettant de déployer au moins 3 services Docker distincts, respectivement pour le **serveur**, le **client web** et la **base de données**.

L'application doit pouvoir être lancée intégralement via docker compose et être fonctionnelle.

3 - Le rendu

Il se fera sous la forme d'une archive au format "zip" contenant le code source, les fichiers annexes (sons, images, etc.), la documentations technique et le manuel utilisateur.

La documentation technique est à destination de professionnels du domaine et contiendra au moins les éléments suivants :

- Informations et éléments à renseigner nécessaires au fonctionnement de l'application,
- Guide de déploiement de l'application,
- Justification du choix des langages et des librairies,
- Diagrammes UML,
- Schéma de la base de données,

Le manuel utilisateur explique lui comment se servir de la solution et présente les différentes fonctionnalités.

Aucun secret (clef d'API, mot de passe, etc.) ne doit être présent dans le rendu. Un secret présent en clair entrainera un malus de points sur la notation en fonction de la criticité de ce dernier.

4 - Le barème

Ce projet est noté sur 500 points avec possibilité d'obtenir 50 points en bonus :

- Documentations : 50 points (une note inférieure à 30 points sur cette partie entraînera un ajournement à ce projet)
 - Documentation technique : 30 points
 - Manuel utilisateur : 20 points
- Qualité de l'interface utilisateur : 10 points
- Qualité de l'expérience utilisateur : 10 points
- Déploiement : 50 points (une note inférieure à 30 points sur cette partie entraînera un ajournement à ce projet)
 - Architecture et abstraction : 30 points
 - Containérisation : 20 points
- Fonctionnalités : 190 points (une note inférieure à 100 points sur cette partie entraînera un ajournement à ce projet).
 - Une fonctionnalité est considérée comme fonctionnelle si elle est implémentée sur le serveur et sur les deux clients.
 - Inscription et connexion : 30 points
 - Connexion utilisateur standard (nom d'utilisateur et mot de passe) : 10 points
 - Connexion via OAuth2 (Facebook, Google, etc.) : 20 points
 - Fonctionnalités générales : 160 points
 - Création d'un groupe : 10 points
 - Création d'une dépense : 40 points
 - Intitulé : 5 points
 - Montant : 5 points

- Date : 5 points
 - Utilisateur ayant réglé la dépense : 5 points
 - Catégorie : 5 points
 - Liste des utilisateurs concernés : 5 points
 - Justificatif (image ou PDF) : 10 points
 - Solde des utilisateurs du groupe : 20 points
 - Équilibrage des comptes automatiques : 30 points
 - Remboursements : 20 points
 - Mise à jour manuelle des comptes : 5 points
 - RIB : 5 points
 - Service de paiement tiers : 10 points
 - Messagerie instantanée : 20 points
 - Export : 10 points
 - Statistiques utilisateurs : 10 points
- Qualité du code : 190 points (une note inférieure à 100 points sur cette partie entraînera un ajournement à ce projet)
 - Le barème item par item est identique à celui des fonctionnalités. Pour un item non réalisé ou complètement dysfonctionnel, la note de qualité de code correspondante sera automatiquement égale à zéro.
 - Les critères appréciés ici sont essentiellement :
 - Structures de données adaptées.
 - Absence de duplication de code inadaptée.
 - Lisibilité du code (cela inclut la cohérence et le sens du nommage des variables et sous-programmes).
 - Facilité de maintenance.
 - Abstraction du code
 - Bonus : jusqu'à 50 points
 - Exemples :
 - L'application est de qualité professionnelle (soin particulier accordé à l'UI/UX, qualité de code excellente, architecture robuste, etc.)
 - L'application est déployée en production et accessible non-localement
 - Des fonctionnalités conséquentes et pertinentes supplémentaires ont été implémentées
 - Malus : jusqu'à l'ajournement du projet
 - Quelques exemples :
 - Secret(s) en clair dans les fichiers, en fonction de la criticité :
 - Criticité mineure (par exemple : clé d'API peu importante, nom d'utilisateur de base de données) : Invalidation de la partie concernée (Qualité du code)
 - Criticité majeure (par exemple : mot de passe de base de données, clé d'API critique) : Ajournement du projet
 - Mot de passe non chiffrés en base de données

[Ajouter un travail](#)

Statut de remise

Groupe	Lille - Groupe 4
Statut des travaux remis	Rien n'a été déposé pour ce devoir
Statut de l'évaluation	Non évalué
Temps restant	7 jours 10 heures restants


Dernière modification	-
Commentaires	► Commentaires (0)

Contactez-nous



Suivez-nous



 Contacter l'assistance du site

Connecté sous le nom « [Séraphin Dubail](#) » ([Déconnexion](#))