Composant du test plan:

* Portee et Apercu

le plan de test couvre les codes javascript presents dans les dossiers que sont basket error-handling, events, promotions et users.

* Strategie de test

les tests seront realisees sur la base des deux ecoles de test logiciel qui sont l’ecole classique et l’ecole londonienne.

* Livrables des tests

Cas de tests, scripts de test, rapports des tests

* Calendrier des tests

Les tests ont été menes sur une periode d’un mois.

* Environnement de test

Les tests ont été realises a l’aide du framework javascipt vitest.

* Approche d’execution des tests

Les cas de test seront exécutés manuellement depuis un terminal et a l’aide de la commande npm run test (si on a besoin de tester un fichier des tests, après avoir entrée la commande fournie appuyer la touche p et entre le nom du fichier)

* Criteres d’entree et de sortie des tests

Une bonne ecriture des codes à tester presque tous les défauts sont corriges

* Risques liees aux test et attenuation

Problèmes d’échecs lors du lancement de certains à cause du manquement de certains éléments dans le code à tester

* Ressources de tests

Étudiants en software testing et framework vitest

* Rapports de tests

1. TEST DES FONCTIONS DU FICHIER BASKET

✓ **calculateTotal (4)**

✓ **Doit me retourner 0 pour un Panier vide**

Objectif du test : S'assurer que la fonction renvoie 0 pour un panier vide.

* Entrée :
* basketItems = []
* Résultat attendu :
* expect(calculateTotal(basketItems)).toBe(0);

✓ **Doit me retourne la somme pour un panier contenant un seul élément**

* Objectif du test : S'assurer que la fonction calcule correctement le total pour un panier avec un seul élément.
* Entrée :
* event = new Event(1, 'ballon', 50, 30, 10, '12/01/24')
* basketItems = [new BasketItem(event, 1)]
* Résultat attendu :
* expect(calculateTotal(basketItems)).toBe(50);

✓ **Doit me retourner la somme pour plusieurs éléments dans le panier**

Objectif du test : S'assurer que la fonction calcule correctement le total pour un panier avec plusieurs éléments.

* Entrée :
* event1 = new Event(1, 'ballon', 50, 30, 10, '12/01/24')
* event2 = new Event(1, 'ballon', 75, 30, 10, '12/01/24')
* basketItems = [new BasketItem(event1, 2), new BasketItem(event2, 1)]
* Résultat attendu :
* expect(calculateTotal(basketItems)).toBe(175);

✓ **Doit me retourner la somme pour des element avec remise**

* Objectif du test : S'assurer que la fonction calcule correctement le total avec une remise.
* Entrée :
* event1 = new Event(1, 'ballon', 50, 30, 10, '12/01/24')
* event2 = new Event(1, 'ballon', 75, 30, 10, '12/01/24')
* basketItems = [new BasketItem(event1, 2), new BasketItem(event2, 1)]
* discount = 20
* Résultat attendu :
* expect(calculateTotal(basketItems, discount)).toBe(155);

✓ **testing showAdverts (2)**

✓ **Doit me retourner true si ' type of user =User'**

* Objectif du test : S'assurer que la fonction renvoie true pour un utilisateur non premium.
* Entrée :
* user = new User(1, "jean") (un utilisateur non premium)
* Résultat attendu :
* expect(showAdverts(user)).toBe(true);

✓ **Doit me retourner true si ' l\'utilisateur n\'est pas premium'**

* Entrée :
* user = new User(1, "jean") (un utilisateur de type User)
* Résultat attendu :
* expect(showAdverts(user)).toBe(true);

✓ **searchBasket (4)**

✓ **Doit me retourner un item avec une correspondance partielle**

- Objectif du test : Vérifier que la fonction renvoie les éléments du panier correspondant partiellement à la requête de recherche.

- Entrée : Recherche avec le terme "ballon".

- Résultat attendu : Un tableau contenant un seul élément, correspondant à l'événement "Ballon d'Or".

✓ **Doit me retourne un item avec une correspondance exacte**

- Objectif du test : Vérifier que la fonction renvoie les éléments du panier correspondant exactement à la requête de recherche.

- Entrée : Recherche avec le terme "Concert de U2".

- Résultat attendu : Un tableau contenant un seul élément, correspondant à l'événement "Concert de U2".

✓ **Doit me retoune un tableau vide pour une recherche sans correspondance**

- Objectif du test : Vérifier que la fonction renvoie un tableau vide lorsqu'aucun élément du panier ne correspond à la requête de recherche.

- Entrée : Recherche avec le terme "Théâtre".

- Résultat attendu : Un tableau vide.

✓ **Doit me retourner un tableau vide pour une recherche dans un panier vide**

- Objectif du test : Vérifier que la fonction renvoie un tableau vide lorsque le panier est vide.

- Entrée : Recherche avec le terme "Ballon" dans un panier vide.

- Résultat attendu : Un tableau vide

✓ **getBasketItem (3)**

✓ **Doit me retourner un BasketItem existant dans le panier**

- Objectif du test : Vérifier que la fonction retourne le `BasketItem` correspondant à l'`Event`.

- Entrée : Un panier contenant deux `BasketItem` et un `Event` qui est présent dans le panier.

- Résultat attendu : La fonction doit retourner le `BasketItem` correspondant à l'`Event`.

✓ **Doit me retourner null si l'item n'est pas dans le panier**

- Objectif du test : Vérifier que la fonction retourne `null` si l'`Event` n'est pas présent dans le panier.

- Entrée : Un panier contenant deux `BasketItem` et un `Event` qui n'est pas présent dans le panier.

- Résultat attendu : La fonction doit retourner `null`.

✓ **Doit me retourner null si le panier est vide**

- Objectif du test : Vérifier que la fonction retourne `null` si le panier est vide.

- Entrée : Un panier vide.

- Résultat attendu : La fonction doit retourner `null`.

✓ **createBasketItem (2)**

✓ **Doit me retourner null si l'element existe deja dans le panier**

* Objectif du test : Vérifier que la fonction retourne null dans ce cas.
* Entrée : Un panier contenant un BasketItem et un Event avec la même quantité.
* Résultat attendu : La fonction doit retourner null.

✓ **Doit créer un nouveau BasketItem**

* Objectif du test : Vérifier que la fonction crée un nouveau BasketItem avec les bonnes informations.
* Entrée : Un panier vide et un Event.
* Résultat attendu : La fonction doit retourner un nouveau BasketItem avec les bonnes informations.

✓ **serializeBasketItemsToJson (3)**

✓ **Doit me retourner un tableau d'items**

* Objectif du test : Vérifier que la fonction retourne bien un tableau.
* Entrée : Un tableau contenant un BasketItem.
* Résultat attendu : La fonction doit retourner un tableau d'items.

✓ **Doit me retourner un tableau d'item ayant une propriete event valide**

* Objectif du test : Vérifier que chaque item du tableau a une propriété event valide.
* Entrée : Un tableau contenant un BasketItem.
* Résultat attendu : Chaque item du tableau doit avoir une propriété event de type object et non null/undefined.

✓ **Doit me retourner un tableau d'item ayant une propriete event valide**

* Objectif du test : Vérifier que chaque item du tableau a une propriété ticketCount valide.
* Entrée : Un tableau contenant un BasketItem.
* Résultat attendu : Chaque item du tableau doit avoir une propriété ticketCount de type number et supérieure à 0.

1. TEST DES FONCTIONS DU FICHIER ACCOUNT

✓ **Purchase (1)**

✓ **Doit me creer un ticket avec des proprietes correctes**

* Description : Vérifier que la classe `Purchase` crée un objet avec les propriétés tickets, eventName et cost initialisées correctement.
* Données d'entrée :
* tickets = 1
* eventName = 'gallerie'
* cost = 200
* Résultat attendu :
* L'objet purchase a les propriétés suivantes :
* purchase.tickets = 1
* purchase.eventName = 'gallerie'
* purchase.cost = 200

✓ **isValidUserName (4)**

✓ Doit me retourner True si username est une chaine de caractere

* Description : Vérifier que la fonction isValidUserName retourne true lorsqu'elle reçoit un nom d'utilisateur valide (une chaîne de caractères non vide).
* Données d'entrée :
* existingUsername = 'john@example.com'
* Résultat attendu :
* La fonction isValidUserName retourne true.

✓ **Doit me retourner False si username est vide**

- Description : Vérifier que la fonction isValidUserName retourne false lorsqu'elle reçoit un nom d'utilisateur vide.

- Données d'entrée :

- existingUsername = ''

- Résultat attendu :

- La fonction isValidUserName retourne false.

✓ **Doit me retourner False pour un usurname null**

* Description : Vérifier que la fonction isValidUserName retourne false lorsqu'elle reçoit un nom d'utilisateur null.
* Données d'entrée :
* existingUsername = null
* Résultat attendu :
* La fonction isValidUserName retourne false.

✓ **Doit me retourner False si username est undefined**

* Description : Vérifier que la fonction isValidUserName retourne false lorsqu'elle reçoit un nom d'utilisateur undefined.
* Données d'entrée :
* existingUsername = undefined
* Résultat attendu :
* La fonction isValidUserName retourne false.

✓ **createAccount (1)**

✓ **Doit me renvoyer InvalidUsernameError si le username est invalide**

* Description : Vérifier que la fonction createAccount lève une erreur InvalidUsernameError lorsque le paramètre username n'est pas valide.
* Données d'entrée :
* existingUsername = 'john@example.com'
* Résultat attendu :
* La fonction createAccount lève une erreur de type "Please enter a valid username".

1. TEST DES FONCTIONS DU FICHIER EVENT

✓ **fonction isSoldOut (2)**

✓ **Doit me retourner false si le nombre de nombre de ticket restant est != 0**

* Objectif du test : Vérifier que la fonction retourne false si le nombre de tickets restants n'est pas égal à 0.
* Entrée : Un événement avec 30 tickets restants.
* Résultat attendu : La fonction doit retourner false.

✓ **Doit me retourner true si le nombre de ticket restant est egale a 0**

* Objectif du test : Vérifier que la fonction retourne true si le nombre de tickets restants est égal à 0.
* Entrée : Un événement avec 0 ticket restant.
* Résultat attendu : La fonction doit retourner true.

✓ **fonction getTagLine (4)**

✓ **Doit me retourner Event Sold Out! si le nombre de ticket restant est egale a 0**

- Objectif : Vérifier que la fonction retourne "Event Sold Out!" lorsqu'il n'y a plus de tickets restants.

- Entrée : Un événement avec 0 ticket restant.

- Résultat attendu : La fonction doit retourner "Event Sold Out!".

✓ **Doit me retourner Hurry only nombre de ticket left! si le nombre de ticket restant est inferieur au minimum de ticket requis**

- Objectif : Vérifier que la fonction retourne un message avec le bon nombre de tickets restants.

- Entrée : Un événement avec moins de tickets restants que le minimum requis (par exemple, 5 tickets restants).

- Résultat attendu : La fonction doit retourner "Hurry only 5 ticket(s) left!".

✓ **Doit me retourner This Event is getting a lot of interest. Don't miss out, purchase your ticket now! si le ticket est populaire**

- Objectif : Vérifier que la fonction retourne le bon message lorsque l'événement est populaire.

- Entrée : Un événement avec des tickets restants et la popularité à "true".

- Résultat attendu : La fonction doit retourner "This Event is getting a lot of interest. Don't miss out, purchase your ticket now!".

✓ **Doit me retourner Don't miss out, purchase your ticket now! si le ticket n'est pas populaire**

- Objectif : Vérifier que la fonction retourne le bon message lorsque l'événement n'est pas populaire.

- Entrée : Un événement avec des tickets restants et la popularité à "false".

- Résultat attendu : La fonction doit retourner "Don't miss out, purchase your ticket now!".

✓ **fonction createEvent (4)**

✓ **Doit me retourner une erreur si nom n'est pas une chaine de caractere ou si la taille du nom est superieur a 200**

✓ **Doit me retourner une erreur si le prix est different d'un nombre ou est inferieur a 0**

✓ **Doit me retourner une erreur si le nombre de ticket disponible est inferieura 1 ou n'est pas un nombre**

✓ **Doit me creer un evenement si toutes les conditions ne sont pas verifiee**

1. TEST DES FONCTIONS DU FICHIER FILTER

✓ **today() (3)**

✓ **Doit me retourner true lorsque la date de l'evenement est aujoud'hui**

Description : Vérifier que la fonction today() renvoie true lorsque la date de l'événement est aujourd'hui.

Données d'entrée :

Un événement avec une date d'aujourd'hui.

Résultat attendu :

La fonction today(event) renvoie true.

✓ **Doit me retourner false lorsque la date de l'evenement n'est pasaujourd'hui**

Description : Vérifier que la fonction today() renvoie false lorsque la date de l'événement n'est pas aujourd'hui.

Données d'entrée :

Un événement avec une date différente d'aujourd'hui.

Résultat attendu :

La fonction today(event) renvoie false.

✓ **Doit me retourner false lorsque la date de l'evenement est avant aujourd'hui**

Description : Vérifier que la fonction today() renvoie false lorsque la date de l'événement est avant aujourd'hui.

Données d'entrée :

Un événement avec une date antérieure à aujourd'hui.

Résultat attendu :

La fonction today(event) renvoie false.

✓ **next7Days() (5)**

✓ **Doit me retourner true lorsque la date de l'evenement est entre les 7 prochains jours**

Description : Vérifier que la fonction next7Days() renvoie true lorsque la date de l'événement se situe dans les 7 prochains jours.

Données d'entrée :

Un événement avec une date comprise entre aujourd'hui et 7 jours après aujourd'hui.

Résultat attendu :

La fonction next7Days(event) renvoie true.

✓ **Doit me retourner true lorsque la date de l'evnement est le jour 7**

Description : Vérifier que la fonction next7Days() renvoie true lorsque la date de l'événement est le 7e jour à compter d'aujourd'hui.

Données d'entrée :

Un événement avec une date correspondant au 7e jour à compter d'aujourd'hui.

Résultat attendu :

La fonction next7Days(event) renvoie true.

✓ **Doit me retourner false lorsque la date de l'evenement est a l'exterieur des 7 prochains jours**

Description : Vérifier que la fonction next7Days() renvoie false lorsque la date de l'événement est en dehors des 7 prochains jours.

Données d'entrée :

Un événement avec une date supérieure à 7 jours après aujourd'hui.

Résultat attendu :

La fonction next7Days(event) renvoie false.

✓ **Doit me retourner false lorsque la date de l'evenement est avant les 7 prochains jours**

Description : Vérifier que la fonction next7Days() renvoie false lorsque la date de l'événement est antérieure aux 7 prochains jours.

Données d'entrée :

Un événement avec une date inférieure à aujourd'hui.

Résultat attendu :

La fonction next7Days(event) renvoie false.

✓ **Doit me retourner false lorsque la date de l'evenement est null**

Description : Vérifier que la fonction next7Days() renvoie false lorsque la date de l'événement est null.

Données d'entrée :

Un événement avec une date null.

Résultat attendu :

La fonction next7Days(event) renvoie false.

✓ **next30Days() (5)**

✓ **Doit me retourner true lorsque la date de l'evenement est entre les 30 prochains jours**

- Description : Vérifier que la fonction `next30Days()` renvoie `true` lorsque la date de l'événement se situe dans les 30 prochains jours.

- Données d'entrée :

- Un événement avec une date comprise entre aujourd'hui et 30 jours après aujourd'hui.

- Résultat attendu :

- La fonction `next30Days(event)` renvoie `true`.

✓ **Doit me retourner true lorsque la date de l'evnement est le jour 30**

-Description : Vérifier que la fonction `next30Days()` renvoie `true` lorsque la date de l'événement est le 30e jour à compter d'aujourd'hui.

- Données d'entrée :

- Un événement avec une date correspondant au 30e jour à compter d'aujourd'hui.

-Résultat attendu :

- La fonction `next30Days(event)` renvoie `true`.

✓ **Doit me retourner false lorsque la date de l'evenement est a l'exterieur des 30 prochains jours**

- Description : Vérifier que la fonction `next30Days()` renvoie `false` lorsque la date de l'événement est en dehors des 30 prochains jours.

-Données d'entrée :

- Un événement avec une date supérieure à 30 jours après aujourd'hui.

- Résultat attendu :

- La fonction `next30Days(event)` renvoie `false`.

✓ **Doit me retourner false lorsque la date de l'evenement est avant les 30 prochains jours**

- Description : Vérifier que la fonction `next30Days()` renvoie `false` lorsque la date de l'événement est antérieure aux 30 prochains jours.

- Données d'entrée :

- Un événement avec une date inférieure à aujourd'hui.

-Résultat attendu :

- La fonction `next30Days(event)` renvoie `false`.

✓ **Doit me retourner false lorsque la date de l'evenement est null**

- Description : Vérifier que la fonction `next30Days()` renvoie `false` lorsque la date de l'événement est `null`.

- Données d'entrée :

- Un événement avec une date `null`.

- Résultat attendu :

- La fonction `next30Days(event)` renvoie `false`.

1. TEST DES FONCTIONS DU FICHIER SEARCH

✓ **getEvents (3)**

✓ **Doit me retourner tous les elements lorsque le predicat n'est pas defini**

- Description : Vérifier que la fonction getEvents() renvoie tous les événements lorsque le prédicat n'est pas défini.

- Données d'entrée :

- Une liste d'événements.

- Un prédicat non défini.

- Résultat attendu :

- La fonction getEvents(events, undefined) renvoie la liste complète d'événements.

✓ **Doit me retouner l'evenement qui correspond au predicat**

- Description : Vérifier que la fonction getEvents() renvoie l'événement qui correspond au prédicat.

- Données d'entrée :

- Une liste d'événements.

- Un prédicat qui correspond à un événement de la liste.

- Résultat attendu :

- La fonction getEvents(events, searchPredicate) renvoie un tableau contenant l'événement correspondant.

✓ **Doit me retourner un tableau vide si aucun element ne correspond au predicat**

- Description : Vérifier que la fonction getEvents() renvoie un tableau vide lorsqu'aucun événement ne correspond au prédicat.

- Données d'entrée :

- Une liste d'événements.

- Un prédicat qui ne correspond à aucun événement de la liste.

- Résultat attendu :

- La fonction getEvents(events, searchPredicate) renvoie un tableau vide.

1. TEST DES FONCTIONS DU FICHIER PROMOTION

✓ **fonction calculatePercentageDiscount (2)**

✓ **Doit me retourner currentPrice \* (discount / 100) si currentPrice >= minimumSpend**

✓ **Doit me retourner currentPrice si currentPrice < minimumSpend**

✓ **fonction calculateMoneyOff (2)**

✓ **Doit me retourner currentPrice - discoun si currentPrice >= minimumSpend**

✓ **Doit me retourner currentPrice si currentPrice < minimumSpend**

✓ **fonction generateReferralCode (1)**

✓ **Doit me retourner #FRIEND-#${id}-#${userId} peut importe ce qu'on introduit dans la fonction lors de son appel**

✓ **fonction applyDiscount (4)**

✓ **Doit me retourner la valeur de la fonction calculateMoneyOff si le data.type = MONEYOFF**

✓ **Doit me retourner la valeur de la fonction calculatePercentageDiscount si le data.type = PERCENTAGEOFF**

✓ **Doit me retourner currentTotal si data.type n'est ni MONEYOFF OU PERCENTAGEOFF**

✓ **Doit me retourner currentTotal si data n'est pas valide**

1. TEST DES FONCTIONS DU FICHIER EXCHANGE

✓ **getExchangeRate (2)**

✓ **Doit appeler exchangeRateProvider et renvoyer exchangeRate**

* Description : Vérifier que la fonction getExchangeRate appelle la fonction callExchangeRateProvider avec le bon paramètre.
* Données d'entrée :
* currencyCode = "USD"
* exchangeRate = 1.25
* Résultat attendu :
* La fonction callExchangeRateProvider est appelée avec le paramètre currencyCode.
* La fonction de rappel est appelée avec un objet contenant les bonnes valeurs pour originalCurrency, newCurrency et exchangeRate.

✓ **Doit appeler la fonction callback avec un response correct**

* Description : Vérifier que la fonction getExchangeRate appelle la fonction de rappel (callback) avec les bonnes données.
* Données d'entrée :
* `currencyCode = "EUR"
* exchangeRate = 1.18
* Résultat attendu :
* La fonction de rappel est appelée avec un objet contenant les bonnes valeurs pour originalCurrency, newCurrency et exchangeRate.

1. TEST DES FONCTIONS DU FICHIER EXCHANGERATEPROVIDER

✓ **callExchangeRateProvider (4)**

✓ **Doit me retourner 1.25 si 'currencyCode = USD'**

* Description : Vérifier que la fonction callExchangeRateProvider retourne le taux de change correct pour la devise USD.
* Données d'entrée :
* currencyCode = "USD"
* Résultat attendu :
* La fonction callExchangeRateProvider retourne 1.25.

✓ **Doit me retourner 1.18 si 'currencyCode = EUR'**

* Description : Vérifier que la fonction callExchangeRateProvider retourne le taux de change correct pour la devise EUR.
* Données d'entrée :
* currencyCode = "EUR"
* Résultat attendu :
* La fonction callExchangeRateProvider retourne 1.18.

✓ **Doit me retourner 1.93 si 'currencyCode = NZD'**

* Description : Vérifier que la fonction callExchangeRateProvider retourne le taux de change correct pour la devise NZD.
* Données d'entrée :
* currencyCode = "NZD"
* Résultat attendu :
* La fonction callExchangeRateProvider retourne 1.93.

✓ **Doit me renvoyer l'erreur Currency not supported si currencycode d'est egale a aucune des valeurs suivantes: USD, EUR, NZD**

1. TEST DES FONCTIONS DU FICHIER USERS

✓ **User (1)**

✓ **Doit me creer un utilisateur avec des proprietes correctes**

- Description : Vérifier que la création d'un utilisateur initialise correctement ses propriétés id, username et isPremium.

- Données d'entrée :

- Un identifiant utilisateur userId.

- Un nom d'utilisateur username.

- Résultat attendu :

- L'objet user créé a les propriétés suivantes :

- user.id est égal à userId.

- user.username est égal à username.

- user.isPremium est égal à false.

✓ **userExists (2)**

✓ **Doit me retourner True si username se trouve dans existingUsers**

- Description : Vérifier que la fonction userExists() renvoie true si le nom d'utilisateur donné existe dans la liste des utilisateurs existants.

- Données d'entrée :

- Un nom d'utilisateur existant (existingUsername).

- Résultat attendu :

- La fonction userExists(existingUsername) renvoie true.

✓ **Doit me retourner False si username ne se trouve pas dans existingUsers**

- Description : Vérifier que la fonction userExists() renvoie `false` si le nom d'utilisateur donné n'existe pas dans la liste des utilisateurs existants.

- Données d'entrée :

- Un nom d'utilisateur non existant (nonExistingUsername).

- Résultat attendu :

- La fonction userExists(nonExistingUsername) renvoie false.