

Vous avez décidé de représenter chaque tuile sur votre chevalet par un Component :

Voici une partie du service :

Votre chevalet est représenté par TileRackComponent. Voici le Component et le gabarit à compléter :

Pouvez-vous quand même utiliser ce service dans votre projet dans son état actuel ?

Vous avez un Service qui possède un attribut de type Subject<string> qui est utilisé pour communiquer avec plusieurs de vos Composants qui s'y abonnent à travers la méthode subscribe(). Pourquoi est-il important de se désabonner avec la méthode unsubscribe() de l'Observable à un moment donné du cycle de vie de chaque Composant ?

- a. Un abonnement non terminé pourrait causer une fuite de mémoire. ✓
 - b. Un Observable peut être observé par un seul Observer à la fois. Le désabonnement permet aux autres Observers de fonctionner.
 - c. La méthode unsubscribe() n'est pas nécessaire puisque RxJS est capable de gérer le désabonnement automatiquement pour vous.
 - d. La librairie RxJS force l'utilisation des méthodes subscribe() et unsubscribe() pour les Observables.
- Voici un extrait du code du gabarit HTML de votre classe GameCarouselComponent qui possède comme enfants des instances de GameCardComponent. Le tableau gameCards possède les jeux à afficher (1 à 4) sous la forme d'un objet GameCard qui contient tous les attributs nécessaires pour l'affichage.
- GameCardComponent possède un seul attribut public qui est : @Input() gameCard: GameCard
- ```
<div id="game-container" *ngIf="gameCards.length > 0">
 <div id="game-container-2" *ngFor="let gameCard of gameCards">
 <app-game-card gameCard="gameCard"> </app-game-card>
 </div>
</div>
```

a) Votre collègue vous demande pourquoi est-ce qu'il y a 2 éléments <div> qui englobent <app-game-card>. Que lui répondez-vous ? Est-ce qu'il y a une manière plus simple qui permet d'avoir le même comportement ? On ne peut pas placer 2 directives structurales sur le même élément, \*ngFor peut aller directement sur <app-game-card>. Les 2 divs ne sont pas nécessaires.

b) Le compilateur d'Angular lance une erreur lors de la transpilation de la ligne 3 du gabarit. Pourquoi ? Le compilateur lance une erreur car le code a été mal écrit, il aurait fallu que ce soit <app-game-card [gameCard]="gameCard"> </app-game-card>

Les tests du côté client utilisant les librairies Jasmine et Karma sont toujours exécutés dans un ordre aléatoire d'un lancement des tests à un autre. Pourquoi ?

- a. Ceci n'a aucun impact sur les tests exécutés et leurs résultats.
- b. Ceci permet d'exécuter les tests plus rapidement.
- c. Ceci est une limitation du fonctionnement de l'outil Karma.
- d. Ceci permet de détecter la présence de dépendances entre les tests unitaires. ✓

Voici le constructeur du composant PlayAreaComponent qui utilise la classe GameService ainsi que la configuration de ses tests unitaires.

```
constructor(private readonly gameService: GameService) {}

// play-area.component.spec.ts
beforeEach(async () => {
 gameSpy = jasmine.createSpyObj('GameService', ['handleClick', 'blinkDifference', 'playSound']);
 TestBed.configureTestingModule({
 declarations: [PlayAreaComponent],
 providers: [{ provide: GameService, useValue: gameSpy }],
 }).compileComponents();
});
```

a) Est-ce que l'utilisation d'un SpyObj pourrait être nécessaire dans la configuration? Justifiez votre choix.

a) Oui, l'utilisation d'un SpyObj pourrait être nécessaire dans la configuration. Ainsi on pourrait utiliser mock afin de se débarrasser des dépendances entre ses tests. On veut tester uniquement le comportement de PlayAreaComponent ainsi que ses appels aux méthodes de GameService sans pour autant toutes les tester ce qui est possible avec l'utilisation de SpyObj.

b) Suite à l'exécution des tests de PlayAreaComponent, vous obtenez à plusieurs reprises le message d'erreur suivant dans la console: ERROR: 'NG0304: 'app-differences-area is not a known element:'.

Quelle est la raison de ce message ?

Selon moi, il s'agit d'un problème d'import, il se peut que app-differences-area soit utilisé par PlayAreaComponent, et si c'est le cas alors il faudrait l'importer et le déclarer par la suite dans declarations.

Vous êtes responsable de faire la revue de code pour la fonctionnalité de dessin avec crayon pour le Sprint 2 d'un autre membre de votre équipe. Voici un extrait du code à revoir. Identifiez les problèmes de qualité avec ce code. Vous pouvez assumer que le code est fonctionnel et que les fonctions et classes référencées existent.

```
const TRUE = true;
const FALSE = false;
const ONE = 1;
const noir = '#000000';

@Injectable({ providedIn: 'root' })
export class PencilService {
 canRedo: boolean = FALSE;
 canUndo: boolean = TRUE;
 public fill = TRUE;
 public circleDiameter: number = ONE;
 Color: any = noir;
 mouseDown: boolean = false;

 private isOnCanvas(event): boolean {
 if (this.drawingService.isOnCanvas(event.offsetX, event.offsetY)) {
 return true;
 } else {
 return false;
 }
 }

 constructor() { this.drawingService = new DrawingService('crayon') }
 // obtenir la taille
 get color() { return this.Color }
 set color(c: string) { this.Color = c }
}
```

- Utilisation du français et de l'anglais, le code est majoritairement écrit en anglais or on a une constante qui s'appelle noir, il aurait fallu écrire black
  - on assigne une couleur noir qui a déjà été déclaré avant a Color, il aurait été préférable de faire Color: any = '#000000'
  - setColor prend pour parametre une variable c or on ne sait pas ce que cela veut dire, ainsi il faudrait changer le nom de la variable pour que cela soit plus instructif
  - pour la méthode onMouseMove(), il n'est pas nécessaire de laisser les lignes 25, 26 et 27. Ainsi a supprimer
  - Déclaration des constantes TRUE et FALSE qui ne sont jamais utilisés donc supprimer ces constant
  - Déclaration d'une constante ONE qui équivalait au chiffre 1 et 1 n'est pas un nombre magique donc faire directement public circleDiameter: number = 1
- fill et circleDiameter sont ici mis en attribut public mais préférable de les mettre en privé.

Lorsqu'on crée une demande fusion (Merge Request) sur GitLab, il est possible de bloquer la fusion tant qu'un pipeline qui lui est associé n'est pas exécuté au complet sans erreurs avec l'option "Pipelines must succeed".

Donnez un avantage et un désavantage de l'utilisation de cette option sur votre processus de travail.

L'avantage de l'option "Pipelines must succeed" est qu'elle assure que le code ne sera fusionné que si tous les tests passent, ce qui contribue à maintenir la qualité du code. L'inconvénient est que cela peut ralentir le processus de fusion, surtout si les pipelines sont longs, même pour des changements qui n'affectent pas la logique du code, comme les modifications de CSS.

L'étape "install" de votre pipeline automatisée sur GitLab exécute la commande npm ci. Pourquoi utiliser cette commande plutôt que npm install ?

- a. npm ci s'assure d'installer les versions exactes des dépendances du projet et toujours reproduire le même environnement, ce qui n'est pas garanti avec npm install. ✓
- b. npm ci veut dire Console Install et est simplement un alias de npm install.
- c. Il y a une erreur dans la configuration du pipeline : npm install devrait être utilisée plutôt que npm ci.
- d. npm ci veut dire Continuous Integration et doit donc être utilisé dans un processus automatisé.
- e. Il n'y a pas de différence entre les commandes et les 2 peuvent être utilisées de manière interchangeable.

Voici l'implémentation de votre ChronometerService qui gère la minuterie dans vos parties. Ses 2 méthodes sont appelées au début et à la fin de chaque partie et l'attribut nbElapsedSeconds est affiché à l'écran dans votre GamePageComponent. La fonction interval est un Observable qui émet à chaque seconde (1000ms). Quelle sera la valeur initiale du temps affichée lorsque vous débutez une 2e partie après avoir complété une partie avant oris 30 secondes ?

```
@Injectable({ providedIn: 'root' })
export class ChronometerService {
 public nbElapsedSeconds: number = 0;
 private chronometer: Subscription = new Subscription();
 public startChronometer(): void {
 this.chronometer = interval(N_MS_IN_SECONDS).subscribe(() => {
 this.nbElapsedSeconds++;
 });
 }
 public stopChronometer(): number {
 if (this.chronometer) this.chronometer.unsubscribe();
 return this.nbElapsedSeconds;
 }
}
```

- a. 30 secondes ✓
- b. Aucune de ses réponses
- c. undefined
- d. 30 secondes + le temps écoulé entre la fin de la 1re partie et le début de la 2e partie
- e. 0 secondes

Parmi les énoncés suivants, lesquels sont erronées ?

- a. L'utilisation de Services est obligatoire dans un projet Angular. ✓
  - b. L'utilisation d'un Service élimine le besoin d'avoir un lien parent/enfant entre les Composants. ✓
  - c. Les décorateurs @Input/@Output permettent le passage de valeurs primitives seulement, contrairement aux Services qui permettent la communication d'objets complexes. ✓
  - d. La syntaxe [{maVariable}] est la seule manière de faire une liaison bi-directionnelle entre un Gabarit et le code TS de son Composant.
  - e. Un Component est identifié seulement par son attribut 'selector' dans le gabarit HTML d'un autre Component.
  - f. La directive \*ngIf ne peut pas être placée sur le même élément HTML que la directive \*ngFor.
- La création d'une nouvelle fiche de jeu dans votre système se fait à travers une requête POST sur la route /games/ et les informations de la fiche dans le corps de la requête. Vous avez décidé de ne pas permettre la création de 2 fiches avec le même nom. Lorsqu'un tel cas arrive, le code de retour dans la réponse HTTP est : 404 Not Found. Sémantiquement, quel est le meilleur code de réponse dans une telle situation ?

- a. 500 Internal Server Error
- b. 204 No Content
- c. 409 Conflict ✓
- d. 404 Not Found
- e. Aucune de ses réponses
- f. 400 Bad Request

Vous êtes responsables de l'implémentation de la logique du système de détection de différences entre 2 images pour le Sprint 1 et vous avez trouvé un service en ligne qui permet de générer les différences entre 2 images avec un rayon d'élargissement. Vous voulez l'utiliser pour générer les images de différences pour vos jeux. Votre équipe a déjà mis en place une communication utilisant Socket.IO pour la logique de jeu entre votre site web et votre serveur. Cependant, le serveur du service trouvé n'utilise pas Socket.IO, mais plutôt l'implémentation native de WebSocket pour sa communication. Pouvez-vous quand même utiliser ce service dans votre projet dans son état actuel ?

- a. Oui, mais vous ne pouvez pas utiliser les fonctionnalités supplémentaires que Socket.IO offre par-dessus WebSocket.
- b. Non, votre serveur ne peut pas communiquer avec un autre serveur, seulement avec des clients.
- c. Oui, mais vous devez établir une connexion supplémentaire en utilisant le protocole WebSocket seulement.
- d. Non, le client et le serveur doivent utiliser Socket.IO pour une communication valide.
- e. Oui, Socket.IO utilise le protocole WebSocket pour communiquer, donc les 2 systèmes sont compatibles.
- f. Oui, mais vous devez contacter le service à travers votre client puisque WebSocket est un protocole client-serveur.

Considérez le code suivant qui gère le téléversement et l'affichage d'une image dans la Vue de création.

```
async imageUpload(e: Event, index: number): Promise<void> {
 let f = (e.target as HTMLInputElement).files.item(0) as File;
 case 1: {
 this.drawImage(this.modCnvs.nativeElement, f);
 break;
 }
 const check = !(this.imagesService.isValidImage(f));
 if (!check) {
 this.displayImageError();
 return;
 }
 case 2: {
 this.drawImage(this.originalCanvas.nativeElement, f);
 this.drawImage(this.modCnvs.nativeElement, f);
 break;
 }
 switch (index) {
 case 0: {
 this.drawImage(this.originalCanvas.nativeElement, f);
 break;
 }
 }
}
```

a) Identifiez les erreurs de qualité de code présentes. Considérez que les autres fonctions appelées sont bien implémentées et ne causent pas de bogues.

Voici une liste d'erreurs de qualité de code :

- La conversion forcée de 'e.target' et du résultat de l'appel 'item(0)' sans vérification.
- La double négation de check lignes 4 et 5.
- La fonction négative isValidImage devrait s'appeler isValidImage et retourner si une image est valide. On peut inverser le résultat de l'appel si besoin.
- Le nom de la variable 'f' n'est pas descriptif de ce que la variable contient.
- Le nom de la variable 'check' n'est pas descriptif non plus.
- L'indentation des cases du switch n'est pas faite.
- Les accolades autour de chaque case du switch sont inutiles.

b) Votre collègue considère que le switch/case présente un problème de duplication de code et propose le code suivant. Est-ce que vous considérez que sa solution est acceptable et améliore la qualité de votre code ?

```
if (index%2 === 0) {
 this.drawImage(this.originalCanvas.nativeElement, f);
}
if (index == 2) {
 this.drawImage(this.modCnvs.nativeElement, f);
}
```

Ce nouveau code n'est pas une meilleure solution car :

- 1. Il ne fonctionne pas pour le cas où index est égal à 1.
  - 2. Le fait d'utiliser l'opérateur modulo ici diminue la lisibilité du code.
  - 3. Il faudrait remplacer le double égal par un triple égal pour valider que le type de index est vraiment number.
- Une fois qu'une demande de fusion (Merge Request) a été revue et approuvée par un membre de l'équipe, qui devrait être responsable de l'intégration du code dans la branche de destination ?
- a. Seulement le membre d'équipe ayant implémenté la majorité du code.
  - b. Le ou les membre(s) désigné(s) par une convention quelconque dans l'équipe.
  - c. Aucune de ses réponses.
  - d. N'importe quel membre de l'équipe.
  - e. Seulement un membre de l'équipe qui n'est pas l'auteur ou l'évaluateur de la demande fusion.
  - f. Seulement le membre d'équipe ayant révisé la majorité du code

a) Voici un extrait du code du gabarit HTML de votre QuestionBuilderComponent qui est utilisé dans la création d'un jeu-questionnaire :

```
<app-question *ngFor="let question of questions; let x = index" *ngIf="questions.length !== 0"
[question]="question" [index]="x + 1" (click)="handleSelect(Sevent)"></app-question>
Voici un extrait du constructeur de ce même composant :
communicationService: CommunicationService;
questions : Question[];
constructor(){
 this.communicationService = new CommunicationService();
 ...
}
```

Identifiez au moins 2 problèmes présents dans ces extraits. Expliquez chaque problème et proposez une solution. Considérez que tout service ou composant externe référencé est fonctionnel.

- 1. Il faut retirer \*ngIf et pas le mettre sur un élément parent :  
2. \*ngFor gère déjà les tableaux vides donc même pas besoin du \*ngIf
  - b) Voici la configuration des tests et un test unitaire de votre AnswerComponent qui soumet une réponse choisie au serveur. Est-ce que cette approche est correcte ? Si oui, pourquoi ? Si non, que devriez-vous changer ?  
Non le test est mal configuré et avec cette configuration on utilise la vraie instance du SocketClientService. Faudrait isoler le composant en créant un mock du service avec SpyObj et l'ajouter aux providers.
- Quel est l'objectif principal d'un stand up dans l'approche Scrum ?
- a. Revoir et mettre à jour le calendrier du projet
  - b. Identifier des obstacles au travail des membres de l'équipe
  - c. Assigner des tâches aux membres de l'équipe
  - d. Faire une revue de code rapide des nouvelles fonctionnalités