

Question 1 : Gestion de projet (2 points)

15/2

Quelques semaines après la fin de projet 2, l'entreprise PolyAQ, agréablement surprise par vos réalisations durant la session, vous contacte afin d'avoir votre expertise pour la réalisation de son projet! Voici les informations que l'on vous donne :

Agile ✓

Tdd ✓

CI - CD X

Autre ✅

- Le projet est proposé par une banque et a donc déjà très précisément spécifié les requis.
- Le projet est une application Web complète (*full-stack*). Vous devez donc développer un client et un serveur qui communiqueront ensemble.
- Pour la connexion et d'autres services de base, votre serveur doit utiliser les API de la banque afin de ne pas avoir à dupliquer des informations de clients (par exemple, un utilisateur qui a déjà un compte dans cette banque n'aura pas à créer un nouveau compte).
- Le client s'attend à un système robuste et veut en être assuré.
- Pour certaines parties critiques, le client fournit même un document spécifiant les limites attendues du système.
- L'échéance du projet est tout de même assez serrée.
- Finalement, le client exige de pouvoir avoir accès au produit en tout temps, et ce dès la fin de la première semaine de travail.

Fier de la confiance qu'ils vous donnent et grandi des apprentissages de projet 2, vous répondez à la demande et vous vous lancez dans la description de l'organisation d'un tel projet! Justifiez chacun de vos choix.

. Nous allons réaliser ce projet selon une méthodologie agile sur plusieurs sprints.

Les développeurs vont prévoir le travail à faire pour chaque sprint avant le début du projet,

afin de rendre l'équipe plus productive. → et assurer respect des délais

Pour l'organisation de notre travail nous utiliserons Trello où chaque → tâche à réaliser est bien décrite dans une carte. Les cartes seront regroupées par sprint.

Le but de cela est ~~de~~ d'avoir une ~~une~~ représentation visuelle de l'avancement

Espace additionnel à la page suivante.

du projet. Nous allons aussi utiliser git comme logiciel de contrôle de gestion. Nous allons commencer par un casse-tête commun et nous allons créer une branche pour chaque fonctionnalité à développer, cela nous permettra de travailler sur différentes fonctionnalités simultanément et éviter de rencontrer des conflits. Pour ce qui est du développement, l'équipe suivra une approche très minutieuse concernant l'assurance qualité, pour faciliter la révision de code et sa réutilisation. Nous utiliserons aussi une approche de développement piloté par les tests, ce qui permettra d'assurer que notre code fonctionne comme il est censé l'être et que les nouvelles fonctionnalités, ne brisent pas les anciennes. Finalement, pour ce qui est des technologies, nous allons utiliser MEAN stack, dont qu'on maîtrise bien, et que nous avons de l'expérience avec.

Nombre de
logiques par
unité doit être
minimal, pour
trouver les
erreurs facilement.

Question 2 : Client (4 points) 3/4

Le Slack du cours ayant dépassé la limite des 10 000 messages, il devient difficile pour les chargés de se souvenir de toutes les demandes ayant été faites au fil du cours. Vous, étudiant assidu et conscientieux, avez pris comme mission de concevoir une petite application Angular permettant de soumettre des demandes de recorrection de la part des équipes de projet 2. Un squelette de code vous est fourni et vous devez compléter les parties manquantes (identifiées par un chiffre allant de 0 à 9) directement dans le code.

Les lignes " _____ " représentent les endroits où vous devez répondre (le nombre de lignes n'est pas nécessairement indicatif de la longueur de la réponse). Notez que vous ne pouvez pas changer la signature des fonctions fournies.

Spécifications demandées :

- Le numéro du sprint (1) ainsi que le numéro de votre équipe (2) doivent être implémentés à l'aide d'un menu déroulant (*dropdown*). La liste des équipes est disponible dans le tableau teams du Component.
- Le bouton "Soumettre la demande" doit envoyer le formulaire (3) en utilisant la méthode sendRecorrection qui retourne une promesse. Lorsque la promesse d'envoi est résolue, l'élément "p" indiquant que la demande a été soumise doit s'afficher (8).
- Vous devez aussi implémenter une liste de toutes les demandes de recorrection (5). Chaque élément de la liste doit comporter le numéro de sprint, le numéro d'équipe ainsi que l'argumentaire (6).
- Cette liste est rafraîchie avec le bouton "Rafraîchir les demandes" (4). La méthode getAllDemands du service injecté (7) permet d'obtenir une promesse contenant toutes les demandes (9).
- Si la demande est traitée, le fond de l'élément doit être mis à lightgreen en appliquant la classe answered (0). Notez que ce changement ne doit pas être fait en TypeScript.

Vous devez compléter le code du style dans le fichier CSS (0).

Vous devez compléter le code du formulaire dans le fichier HTML (1, 2, 3, 4, 5, 6).

Vous devez compléter le code du Component Angular dans le fichier TS (7, 8, 9).

Un visuel de l'application vous est fourni à la page suivante (Figure 1).

LOG2990

Examen final

Hiver 2019

Demande de recorrection

Ce formulaire de recorrection vous permet d'envoyer une demande aux chargés de laboratoire pour un dit sprint. Rentrez les informations...

Numéro du sprint

Numéro de votre équipe

Vos arguments

Votre note pourra être revisée à la baisse

La demande a été soumise!

Liste de toutes les recorrections demandées

Sprint: 4	Team: 100	Arguments: Ça ne compile pas sous Linux...
Sprint: 1	Team: 100	Arguments: ...

Figure 1 Page d'accueil de l'application

Fichier CSS :

```

/* 0. À COMPLÉTER POUR METTRE LA COULEUR DE FOND */
main {
    background-color: lightgreen; X ansuocel
}
ul li.demand {
    list-style: none;
    margin-bottom: 2em;
    border: dashed 2px black;
}

```

-0.25

Fichier HTML :

```

<main>
  <h1>Demande de recorrection</h1>
  <p>Ce formulaire de recorrection vous permet d'envoyer une demande aux chargés
    de laboratoire pour un dit sprint. Rentrez les informations...</p>
  <form>
    <label for="sprintNumber">Numéro du sprint
      <select name="sprintNumber" [(ngModel)]="sprintNumber">
        <!-- 1. À COMPLÉTER -->
        <li> 1 </li>
        <li> 2 </li>
        <li> 3 </li>
        <li> 4 </li>
      </select> X (option)
    </label>
    <label for="teamNumber">Numéro de votre équipe
      <!-- 2. À COMPLÉTER -->
      <select name="teamNumber" [(ngModel)]="teamNumber">
        <option *ngFor="let m of teams" value="{{m}}> {{m}} </option>
      </select>
    </label>
    <label for="complaints">Vos arguments
      <textarea name="complaints" required rows="10" cols="80"
        [(ngModel)]="complaintArguments"></textarea>
    </label>
    <!-- 3. À COMPLÉTER : BOUTON AVEC LES BONS ATTRIBUTS -->
    <button type="submit" (click)="sendForm($event)">Soumettre
      la demande</button>
    <small>Votre note pourrait être revisitée à la baisse</small>
  </form>
  <p *ngIf="submitted">La demande a été soumise!</p>
  <h1>Liste de toutes les recorrections demandées</h1>
  <!-- 4. À COMPLÉTER : BOUTON AVEC LES BONS ATTRIBUTS -->
  <button type="button" (click)="refreshDemands()">Rafraîchir
    les demandes</button>
  <ul>
    <!-- 5. À COMPLÉTER : LISTE DES DEMANDES -->
    <li *ngFor="let d of allDemands" [ngClass]="{'answered': d.answered}" class="demand">
      <!-- 6. À COMPLÉTER : LISTE DES INFORMATIONS COMME SUR LA FIGURE 1 -->
      <h2> {{d}} </h2>
    </li>
  </ul>
</main>

```

-0.25

-0.25

-0.25

✓ Interpolation OK
 ✗ HTML : h2 est pas trop mal,
 mais ça va afficher
 [Object Object]
 (sinon il faudrait utiliser
 le JSONPipe)

Fichier TS :

```

import { Component } from '@angular/core';
@Component({})
export class AppComponent {
  public teams: number[] = [
    100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114,
    115, 116, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213,
    214, 215, 216, 217, 218 ];
  public sprintNumber: number;
  public submitted: boolean = false;
  public teamNumber: number;
  public complaintArguments: string;
  public allDemands: IRecorrectionDemand[] = []; // Déjà initialisé
  // 7. À COMPLÉTER
  constructor(private demandeRecorrectionService: DemandeRecorrectionService) {
    private buildRecorrectionDemand(): IRecorrectionDemand {
      return {
        sprint: this.sprintNumber,
        team: this.teamNumber,
        complaintArguments: this.complaintArguments,
        answered: false,
      };
    }
    public sendForm($event: Event): void {
      // 8. À COMPLÉTER EN UTILISANT LE SERVICE
      demand: IRecorrectionDemand = buildRecorrectionDemand();
      demandeRecorrectionService.sendRecorrection(
        demand).catch(res: boolean) => {
          if (res) { this.submitted = res ?? this.submitted = true; } else { this.submitted = false; }
      }
    }
    public refreshDemands(): void {
      // 9. À COMPLÉTER
      demandeRecorrectionService.getAllDemands().
      catch(res: IRecorrectionDemand) => {
        this.allDemands = res;
      }
    }
  }
  interface IRecorrectionDemand {
    sprint: number;
    team: number;
    complaintArguments: string;
    answered: boolean;
    id: number;
  }
  export class DemandeRecorrectionService {
    public sendRecorrection(demande: IRecorrectionDemand)
      : Promise<boolean> {...}
    public getAllDemands(): Promise<IRecorrectionDemand[]> {...}
  }
}

```

(3)

Question 3 : Serveur (4 points)

Vous devez implémenter une partie du serveur qui gère les demandes de recorrection. Cette partie permet aux chargés de laboratoire de traiter les différentes demandes envoyées et est accessible seulement sur l'adresse /admin. Les fonctionnalités demandées sont présentées dans le tableau ci-dessous avec leurs routes et leur description. Notez que les numéros d'identification de demandes sont uniques, c'est-à-dire que le même numéro de demande ne sera jamais utilisé pour deux équipes différentes (ni pour une même équipe, évidemment). Pour chaque fonctionnalité, vous devez vous assurer de :

- Utiliser les bons verbes HTTP;
- Retourner les bons codes HTTP;
- Gérer les cas invalides;
- S'il y a lieu, vous assurer que les paramètres en entrée sont valides.

La liste des fonctionnalités est présentée dans le tableau 1.

Tableau 1 Liste des fonctionnalités

Fonctionnalité	Route
Récupérer toutes les demandes sur le serveur (10)	/demands
Récupérer toutes les demandes d'une équipe en particulier (11)	/demands/teams/<id équipe>
Marquer une demande comme répondue (12)	/demands/answer/<id demande>
Supprimer une demande (13)	/demands/<id demande>
Récupérer une demande aléatoire non-répondue parmi toutes les demandes d'une équipe (14)	/demands/random/<id équipe>

- Complétez le code des fonctionnalités (10, 11, 12, 13) du serveur fourni à la page suivante. Pour sauver du temps et de l'espace, vous pouvez implémenter directement vos fonctionnalités dans vos routes. Attention à la duplication de code. (3,5 points)
- Pour la dernière fonctionnalité (14) concernant la récupération d'une demande aléatoire, complétez le code du serveur, mais en utilisant la fonction validateDemand qui permet de valider si une demande est bien remplie avant d'être envoyée aux chargés. Assurez-vous de bien gérer tous les cas possibles. (0,5 point)

Répondez aux deux questions à la page suivante.

Code fourni :

```

import * as express from "express";
import * as bodyParser from "body-parser";
import { IRecorrectionDemand } from "IRecorrectionDemand";

function validateDemand(demand: IRecorrectionDemand): Promise<boolean> { ... }

let app: express.Application = express();

// les demandes sont sauvegardées ici :
let recorrectionDemands: IRecorrectionDemand[] = [];
app.use(bodyParser.json());

```

, res: Response

// 10. À COMPLÉTER

```

app.get("/demands", req: Request, next: NextFunction) => {
    res.send(recorrectionDemands);
}

```

Q.75

// 11. À COMPLÉTER

```

app.get("/demands/:id", req: Request, res: Response, next: NextFunction) => {
    tab: IRecorrectionDemand[] = [];
    id: number = parseInt(req.param("id"));
    for (int i = 0; i < recorrectionDemands.length; i++) {
        if (recorrectionDemands[i].id === id) {
            tab.push(recorrectionDemands[i]);
        }
    }
    res.send(tab);
}

```

Q.75

Suite page suivante.

post / put

// 12. À COMPLÉTER

```

app.use("/demands/answers/:id", req: Request, res: Response,
next: NextFunction) => {
    id: number = parseInt(req.param.id);
    for (int i=0; i < reconnectionDemands.length; i++) {
        if (reconnectionDemands[i].id === id) {
            reconnectionDemands[i].answered = true;
        } else?
    }
}

```

Q. 25

{;

Pas de retour HTTP

// 13. À COMPLÉTER

```

app.delete("/demands/:id", req: Request, res: Response,
next: NextFunction) => {
    id: number = parseInt(req.param.id);
    for (int i=0; i < reconnectionDemands.length; i++) {
        if (reconnectionDemands[i].id === id) {
            reconnectionDemands.splice(i, 1);
        } else?
    }
}

```

Q. 5

{;

Suite page suivante.

// 14. À COMPLÉTER

app.get("/demands/random/:id", req: Request, res: Response,
next: NextFunction) => {
 temp: If ~~recorrectionDemand~~ Demand[] = []; id: number = ~~parseInt~~(req.params.id);
 for (int i=0; i < ~~recorrectionDemands.length(); i++~~) {
 if (~~recorrectionDemands[i].team == i~~) {
 temp.push(~~recorrectionDemands[i]~~)
 }
 }
 random: number = Math.floor(~~temp.length * Math.random()~~);
~~Post. boolean else~~
~~if (validateDemande(temp[random]) {~~
Promise ~~res.send(temp[random]);~~
};
};
app.listen(3000);

(0.5)

Question 4 : MongoDB (1 point)

- a) Une projection avec MongoDB permet d'inclure ou d'exclure certains attributs de la réponse d'une requête. Est-ce que les projections suivantes sont correctes? Si oui, expliquez pourquoi la requête est valide. Sinon, expliquez pourquoi la requête n'est pas valide. Tenez pour acquis que le format des documents sur la base de données est le même que celui de l'interface `IRecorrectionDemand` de la question 2. (0,5 point)

i. `{ projection : { team : 1, sprint : 0 } }`

. Cette projection est valide.
Elle permet d'obtenir toutes les équipes de la collection

ii. `{ projection : {team : 1, sprint : 1, _id : 0 } }`

. Cette projection n'est pas valide car on a choisi de donner un id ~~mores~~ même.
Il faut plutôt utiliser
`{projection: { team: 1, sprint: 1; id: 0 }}`

- b) Écrivez la requête permettant de retrouver toutes les demandes de reparation de l'équipe 301. Triez le résultat en ordre descendant selon le numéro du sprint. (0,5 point)

demandes.find({team: {\$eq:301}}).sort
({sprint:-1})

Question 5 : ThreeJS (2 points)

Pour nous pratiquer avec ThreeJS, nous voulons ajouter un même cube dans deux scènes différentes. Nous avons écrit le code donné à la page suivante. À notre grande surprise, le cube n'a été ajouté qu'à une seule scène (Figure 2).

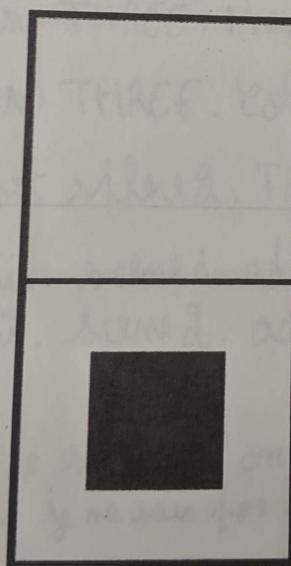


Figure 2 Rendu obtenu

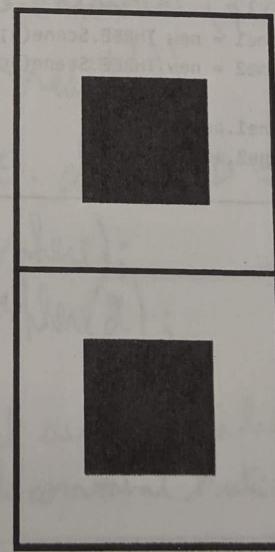


Figure 3 Rendu voulu

Code fourni :

```
public ngAfterViewInit(): void {
    this.renderer1 = new THREE.WebGLRenderer();
    this.div1.nativeElement.appendChild(this.renderer1.domElement);
    this.renderer2 = new THREE.WebGLRenderer();
    this.div2.nativeElement.appendChild(this.renderer2.domElement);
    this.renderer1.setSize(RENDERER_SIZE, RENDERER_SIZE);
    this.renderer2.setSize(RENDERER_SIZE, RENDERER_SIZE);
    this.camera1 = new THREE.Camera();
    this.camera2 = new THREE.Camera();

    this.createScene();
    this.render();
}

public render(): void {
    requestAnimationFrame(() => this.render());
    this.renderer1.render(this.scene1, this.camera1);
    this.renderer2.render(this.scene2, this.camera2);
}

// Début de la partie importante
public createScene(): void {
    // Création de notre cube
    const cube : THREE.Object3D = new THREE.Mesh(new THREE.BoxGeometry(),
        new THREE.MeshBasicMaterial({color: new THREE.Color("green")}));
    this.scene1 = new THREE.Scene();
    this.scene2 = new THREE.Scene();

    this.scene1.add(cube);
    this.scene2.add(cube);
}
```

- a) Quel changement doit-on effectuer dans le code pour obtenir le rendu voulu (Figure 3)?
Écrivez seulement le code nécessaire pour corriger la situation. (1 point)

Il faut cloner le cube l'ajouter à une
scène, et ajouter son clone à
l'autre scène.

- b) En vous basant sur la fonction createScene du code donné ci-dessus, créez une sphère de couleur rouge et ajoutez-la dans les deux scènes. Notez que votre code sera inséré après l'instruction this.scene2.add(cube). (1 point)

```
const sphere : THREE.Object3D = new THREE.  
.Mesh(new THREE.SphereGeometry(),  
new THREE.MeshBasicMaterial({color:  
new THREE.Color("red")});
```

```
const sphere2 : THREE.Object3D = sphere;  
this.scene1.add(sphere);  
this.scene2.add(sphere2);
```

PS: je sais qu'on peut aussi utiliser clone
mais je ne suis pas sûr de comment l'utiliser, merci!

Question 6 : Assurance qualité (3 points)

Le code suivant présente des problèmes de qualité que vous saurez aisément reconnaître et a besoin de réusinage (*refactoring*).

```
app.get("/collection/:id", getItem);

function getItem(req: express.Request, res: express.Response): void {
  const items: Items[] = getAllItems(parseInt(req.params.id));
  if (items.length > 0) {
    const item: Item = items[Math.floor(Math.random() * items.length)];
    const processedItem = processItem(item);
    res.status(200).send(processedItem);
  } else {
    res.status(404).send("No items");
  }
}
```

- Réécrivez le code de manière qu'il n'ait plus de défauts. (1,5 point)
- Sans écrire le code, expliquez comment vous ferez pour tester adéquatement la fonction `getItem()`. (1,5 point)

Répondez aux deux sous-questions à la page suivante.

D

Répondez à la question 6 ici.

a) `app.get ('/collection/:id', req: Request, res: Response, next: NextFunction) => {
 getItem (req, res, next);
}`

0

`function getItem (req: Request, res: Response, next: NextFunction): void {
 const items: Item[] = getAllItems (parseParam (req.params.id));
 if (items.length () > 0) {
 res.status (200). send (randomItem (Math.floor (Math.random ()) + items.
 } else {
 res.status (404). send ("No item");
 }
}`

length()

b) On vérifie que la fonction est bellement également appelée avec l'URI fourni, d'une part. D'autre part, il faut vérifier le bon fonctionnement de la fonction et qu'elle fait ce qu'elle doit faire convenablement.

1

on utilise un `http.ClientSpy`.

1 seule fois)

1.75

Question 7 : Gestionnaire de versions Git (2 points)

a) Quelle est l'utilité d'avoir des tags ? (0,25 point)

On peut retrouver un commit plus facilement } X

b) Expliquez à quoi sert la commande *stash*. Donnez une situation où l'utilisation de cette commande serait pertinente. (0,5 point)Utilisation :Situation :

c) Comment limiter les risques de conflits ? Nommez deux manières. (0,5 point)

- S'assurer que le nombre de lignes est minimal ✓
• Travailler sur des branches différentes et intégrer, en continu (à la fin de chaque tâche). ✓

d) Donnez trois avantages à travailler sur des branches. (0,75 point)

- On peut travailler sur différentes fonctionnalités simultanément et éviter de rencontrer des conflits ✓
- Si il y'a un bugue, il reste sur cette branche et ne touche pas le reste des branches. ✓
- Si on veut retrouver un commit, ça peut aider à le retrouver plus rapidement. ✓

Question 8 : Socket.IO (2 points)

- a) Qu'est-ce qu'une chambre (room) lorsqu'on parle de socket ? (0,5 point)

1.3

C'est un regroupement de plusieurs sockets.

0.25

- b) Citez deux exemples d'utilisation des chambres de Socket.IO. Vous pouvez répondre dans le contexte du projet 2. (0,5 point)

Lorsque deux joueurs sont dans une partie 1V1 créer un room permet de gérer les messages du jeu de cette partie là et éviter que d'autres personnes les reçoivent.

(pour deux
seulement)

0.25

~~On peut mettre tous les sockets dans un seul room,~~ On peut aussi mettre tout les sockets dans un seul room, pour que tout le monde reçoivent les messages de connexion/déconnexion.

- c) On vous donne le code suivant. On vous demande de compléter le code du côté serveur qui permettra de créer la chambre room1. Le code du côté serveur doit établir la connexion avec le socket, écouter sur l'événement et réagir en fonction. (1 point)

```
// Code côté client  
let socket: SocketIOClient.Socket = socketIO('http://localhost:3000/');  
socket.emit('createRoom', 'room1');  
  
// Code côté serveur  
io: SocketIO.Server;  
// À COMPLÉTER
```

```
io.on("connection", (socket: SocketIO.Socket) => {  
    socket.on("createRoom", (msg: string) => {  
        if (msg === "room1") {  
            socket.join("room1");  
        }  
    }  
});
```

1

0.5

LOG2990

Question 9 : Bonus (0,5 point)

Trouvez les 7 différences. Encerclez chaque différence directement sur l'image modifiée (Figure 5).
Pour avoir le bonus, vous devez trouver toutes les différences.

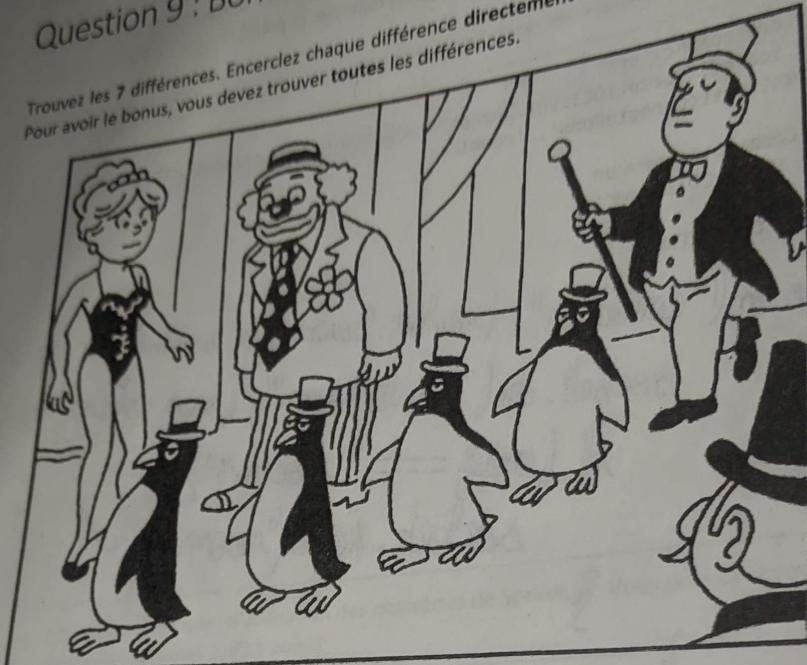


Figure 4 Image originale

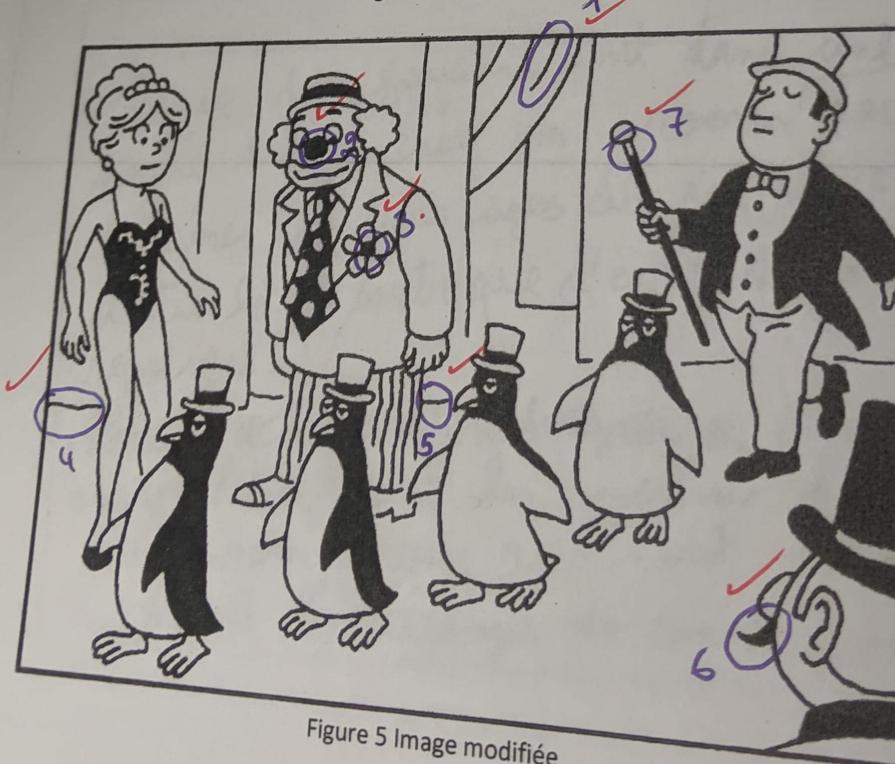


Figure 5 Image modifiée