Nom:_____

Code permanent: _____

(/ 80 pts)

Examen Intra INF5190 - Automne 2022

Directives

- Écrivez vos réponses directement sur le questionnaire.
- Aucune documentation permise à l'exception d'une feuille de notes recto-verso.
- Fermez et rangez votre téléphone cellulaire.
- L'examen comporte 18 questions sur 14 pages.

1. À partir de l'URL suivant, identifiez les éléments demandés. (4pts)

https://aaa:bbb@exemple.com/catalogs/12345/products/?sort=desc&order_by=name

a) Protocole: **https**

b) Domaine (hôte) : **exemple.com**

c) Le(s) paramètre(s) de la requête (*query parameters*) et leur valeur:

sort=desc&order_by=name

d) Chemin d'accès (path) : **/catalogs/12345/products/**

2. Expliquez les différences principales entre les sites dynamiques et les applications monopage en ce qui concerne:

a) la génération des pages HTML de l'application web (2 pts)

Site dynamique: Pages de l'application générées par le serveur.

Monopage: Un seul fichier HTML téléchargé. C'est le code javascript qui modifie le DOM pour générer les différentes pages de l'application.

b) la gestion de la navigation entre les différentes pages (2 pts)

Site dynamique: Navigation => Requête au serveur pour obtenir la nouvelle page

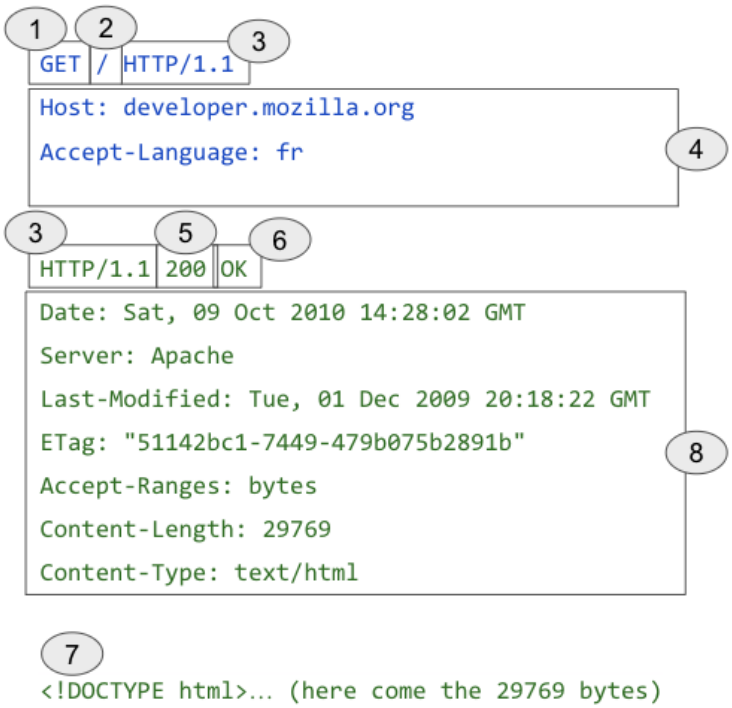
Monopage: Navigation => Le code javascript modifie le DOM pour générer la nouvelle page

Note: pas d'appel d'API nécessaire pour changer de page dans une application monopage.

3. Quelle est la différence entre les paramètres régionaux fr_CA et en_CA? (2pts)

La langue: français Canada vs anglais Canada

4. En utilisant l'image ci-dessous, indiquez le ou les numéros (1 à 7) correspondants aux éléments suivants: (5pts)



a) Corps (body): **7**
b) Protocol et version: **3**
c) Code de statut (status code): **5**
d) Méthode HTTP: **1**
e) En-têtes : **4 et 8**

5. En considérant une architecture en trois couches (présentation, traitement, données) pour une application monopage, indiquez la ou les couches responsables de: (5pts)
-0,5 si manque une couche ou plus pour une ligne

Responsabilité	**Couche(s)**
Stockage des comptes utilisateur	**D**
Logique d'affaire (*business logic*)	**T**
Appels d'API	**P T (appel d'api à partir du javascript et appel d'API entre les services de la couche de traitement)**

Interface utilisateur	**P**
Authentication	**P T D (UI, traitement pour le hash et comparaison, données pour stocker)**

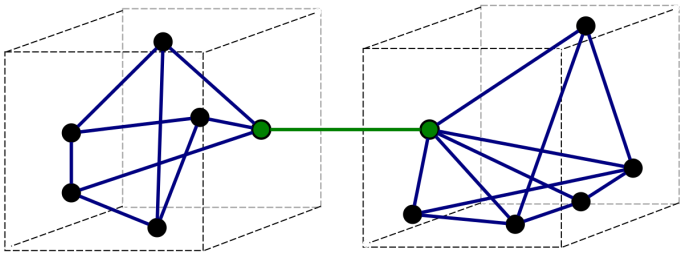
6. Expliquez la différence principale entre une architecture monolithique et une architecture par services. Ensuite, nommez 1 avantage et 1 inconvénient pour chacune des deux architectures. (6pts)

2 pts pour la différence, 1 pt par avantage, inconvénient.

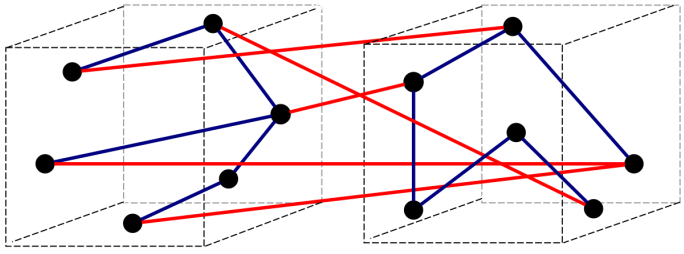
Architecture Monolithique: Un seul élément déployé. Communication interne via fonctions.

Architecture par Service: Plusieurs éléments déployés et interconnectés. Communications via API.

7. Lors de la conception d'un système en utilisant une architecture par services, expliquez dans vos mots les caractéristiques désirables suivantes: (5pts)



a) Good (loose coupling, high cohesion)



b) Bad (high coupling, low cohesion)

Forte cohésion: **Chaque service est responsable d'un ensemble de fonctionnalités qui couvre un domaine d'affaires bien défini. Ces fonctionnalités sont fortement dépendantes les unes des autres.**

Single Responsability Principle. Ce qui change ensemble reste ensemble.

Faible couplage: **Le nombre de dépendances entre les différents services est réduit (essentiellement communication via API et sérialisation). Un changement dans l'implémentation d'un service n'affecte pas (ou très peu) les autres services.**

8. Expliquez comment un serveur peut vérifier si un jeton JWT a été modifié? (2pts)
On peut vérifier si un jeton a été modifié en calculant sa signature (hash(header + . + payload, secret)) et en la comparant avec celle qui est présente dans le jeton reçu. Si les signatures ne sont pas identiques, le jeton a été modifié.

9. Dans la conception d'une application monopage par composants avec Angular, on distingue trois types de composants: les composants de présentation, les composants intelligents et les services. (5pts)

Pour chacune des responsabilités suivantes, indiquez le type de composant qui lui est généralement associé.

Responsabilité	Composant
Afficher une partie de l'interface utilisateur (exemple: une liste de produits)	**P**
Appeler un API backend	**S**
Coordonner plusieurs composants de présentation	**I**
Recevoir des valeurs via des `Input` et émettre des valeurs via des `Output`.	**P**
Appeler les services	**I (S ok, formulation ambigüe)**

10. Pourquoi sérialise-t-on les données lors de la communication entre deux services web? (2pts)
Interopérabilité - deux services utilisant des langages de programmation différents peuvent s'échanger des données.

11. Définissez ce qu'est une insertion partielle en modélisation de données dans un contexte de base de données non relationnelle. Donnez ensuite un exemple d'insertion partielle. (4pts)

Duplication de données. On insère, dans un document, des informations provenant d'un autre document auquel il fait référence. (2pts)

2 pts - pour l'exemple

12. Sérialisation JSON.

a) Étant donné la définition de `Livre` (`record` Java) suivante. Écrivez la représentation JSON équivalente à l'instance `livre` définie ci-bas. (3pts)

```
public record Livre(String titre, int nbPages, String[] auteurs) {}

Livre livre = new Livre("Le bon livre", 123, new String[] { "A.
Long", "B. Short" })
```

```
{
  "titre":"Le bon livre",
  "nbPages":123,
  "auteurs":[
    "A. Long",
    "Cray Ion"
  ]
}
```

b) Étant donné le JSON suivant, définissez le `record` java correspondant et écrivez le code pour créer une instance équivalente à la représentation JSON ci-dessous. (3pts)

```
{
   "nom":"Cray Ion",
   "age":32,
   "livres":[
      {
         "titre":"Le bon livre",
         "nbPages":123,
         "auteurs":[
            "A. Long",
            "Cray Ion"
         ]
```

```
        }
    ]
}
```

```
Livre l = new Livre("Le bon livre", 123,
                    new String[] {"A. Long", "Cray Ion"});

return new Auteur("Cray Ion", 32, new Livre[] {livre});
```

13. Pourquoi utiliser une fonction de hachage lorsqu'on stocke des mots de passe dans une base de données? Comment valide-t-on ensuite le mot de passe reçu lors de la connexion (login) de l'utilisateur? (2pts)

Pour éviter de stocker le mot de passe en clair dans la base de données (confidentialité et contre les attaques).

Comparaison du hash entre le pwd reçu et le pwd dans la BD pour l'utilisateur en question.

Note: on ne fait pas de requête dans la table avec la valeur du hash, mais avec la valeur du nom d'utilisateur.

14. Comment peut-on s'assurer que si un serveur tombe en panne le système sera toujours 100% fonctionnel? (2pts)

Redondance: plus d'instances qu'il en faut pour la charge du système.

Note: Résilience n'implique pas 100% fonctionnel. Fallback ou standby implique un moment où le système risque de n'est pas être fonctionnel à 100%.

Chaos Engineering: Méthodologie de test qui aide à trouver des problèmes, mais n'assure pas que ça fonctionne.

15. Définissez le modèle de données pour une base de données orientée documents pour les entités suivantes: (6pts)

Produit

Un produit possède un nom, une description et une liste de formats de vente.

Format de vente

Un format de vente possède un nom, une description et un prix.

Donnez le nom de la ou des collections nécessaires. Donnez ensuite un exemple de document de type Produit qui possède 2 Formats de vente et un exemple de document de type Format de vente. Optimisez la structure du document représentant les Produits sachant qu'on désire afficher les noms et prix des Formats de vente lorsque l'on affiche les Produits.

Exemple: On définit une collection <nom de la collection> pour les entités de type <nom du type>. Voici un exemple de document pour <nom du type> {...}.

```
2 collections: Produits et Formats

Possiblement Formats comme sous collection de Produits :
Produits/id/Formats

Exemple de Format de vente:

{
  nom: "sac 10kg",
  description: "sac en plastique de 10kg",
  prix: 1000
  id_produit: "abcdef"  // pas considéré dans la correction
}

Exemple de Produit:

{
  nom: "carotte",
  description: "carottes de champ savoureuses",
  formats: [
    {
      id: "12345"  // Nécessaire pour la synchronisation des données
      nom: "sac 10kg",
      prix: 1000
    },
    {
      id: "56789"  // Nécessaire pour la synchronisation des données
      nom: "sac 20kg",
      prix: 2000
    }
  ]
}
```

16. Définissez un API de type RESTfull pour effectuer les opérations suivantes pour un système de location de jeux de société: (8pts)

- Charger tous les jeux (disponibles ou non)
- Ajouter un jeu
- Modifier un jeu
- Supprimer un jeu
- Charger les jeux créés par un éditeur spécifique
- Charger tous les commentaires à propos d'un jeu spécifique
- Ajouter un commentaire à propos d'un jeu spécifique
- Emprunter un jeu spécifique

Utiliser *{nom_variable}* pour indiquer une variable dans un URL.

URL	Méthode HTTP	Paramètres de requête (si applicable)
/jeux	**GET**	
/jeux	**POST**	
/jeux/{id_jeux}	**PUT**	
/jeux/{id_jeux}	**DELETE**	
/jeux/ **ou** **/editeurs/{id_editeur}/jeux**	**GET**	**?editeur = id_editeur**
/jeux/{id_jeux}/commentaires/	**GET**	
/jeux/{id_jeux}/commentaires/	**POST**	
/jeux/{id_jeux}/emprunter **ou** **/emprunter**	**POST**	**Pas seulement modifier l'enregistrement du jeu, probablement une transaction avec validation et info utilisateur.**

17. Étant donné le code java ci-dessous, indiquez la réponse du serveur pour les appels suivants. (6pts)

```java
@RestController
public class ApiController {
    private String nom = "Alice";

    @GetMapping("/")
    String get(@RequestParam("nom") Optional<String> nom) {
        final String n = nom.isPresent() ? nom.get() : this.nom;
        return "Bonjour " + n;
    }

    @GetMapping("/{nom}")
    String get(@PathVariable("nom") String nom) {
        return "Salut " + nom;
    }

    @PostMapping("/nom")
    void post(@RequestBody() String nom) {
        this.nom = nom;
    }
}
```

<u>Les appels sont exécutés dans l'ordre.</u>

Requête	Réponse
GET /	Bonjour Alice
GET /Bob	Salut Bob
POST /nom (le corps de la requête est Alex)	
GET /	Bonjour Alex
GET /Julie?nom=Jeanne	Salut Julie

18. En utilisant les deux composants Angular ci-dessous et en supposant que le composant affiché par la route actuelle est le `PageComponent`. (6pts)

a) Si on entre la valeur 'abc' dans l'*input* et qu'on clique sur le bouton Soumettre, inscrivez le texte qui sera affiché à l'intérieur de la balise `<p>` du `PageComponent`.

ABC - INF5190

1 pt pour la valeur,

1 pt pour uppercase

b) Le code affiche des messages dans la console. Indiquez la séquence des messages qui s'afficheront dans la console si on charge la page, on écrit 'abc' dans l'*input*, puis on appuie sur le bouton `Soumettre`.

PageComponent - ngOnInit

FormulaireComponent - click

PageComponent - surSoumission

FormulaireComponent - reset

```
@Component({
 selector: 'app-page',
 template: `
   <app-formulaire
     [suffix]="suffix"
     (soumission)="surSoumission($event)"
   ></app-formulaire>
   <p>{{ valeur | uppercase }}</p>
 `,
})
export class PageComponent implements OnInit {
 @ViewChild(FormulaireComponent)
 formulaire: FormulaireComponent | null = null;

 suffix = ' - inf5190';
 valeur = '';

 constructor() {}

 ngOnInit(): void {
   console.log('PageComponent - ngOnInit');
 }

 surSoumission(texte: string) {
   console.log('PageComponent - surSoumission');
   this.valeur = texte;
   this.formulaire?.reset();
 }

 click(texte: string) {
   console.log('PageComponent - click');
   this.valeur = texte;
   this.formulaire?.reset();
 }
}
```

```
@Component({
 selector: 'app-formulaire',
 template: `
   <form [formGroup]="form" (ngSubmit)="click()">
     <input type="text" formControlName="texte" />
     <button type="submit">Soumettre</button>
   </form>
 `,
})
export class FormulaireComponent implements OnInit {
 form = this.formBuilder.group({
   texte: [''],
 });

 @Input()
 suffix = '';

 @Output()
 soumission = new EventEmitter<string>();

 constructor(private formBuilder: FormBuilder) {}

 ngOnInit(): void {}

 click() {
   console.log('FormulaireComponent - click');
   if (this.form.valid && this.form.value.texte) {
     this.soumission.emit(this.form.value.texte + this.suffix);
   }
 }

 reset() {
   console.log('FormulaireComponent - reset');
   this.form.reset();
 }
}
```