|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | **Informaticien/-ne CFC**  Travail pratique individuel 2023 (TPI) |  | |
|  | |

Modèle de rapport v1.4

**Candidat No**

**145341**

Nom du candidat : Killian Pasche

Attention : Dernier délai de remise du rapport durant la réalisation du TPI le 18.06.2021 à 17h00 hormis la présentation (ce texte doit être effacé).

**Futnet Single Master**

### Sommaire

Résumé du rapport du TPI 2

1 Les grandes lignes du projet 3

1.1 Analyse de la situation initiale 3

1.2 Analyse de l’état désiré 3

1.3 Cahier des charges / exigences du système 3

1.4 Organisation du projet 4

1.5 Technologies 4

1.6 Test technologique 5

2 Analyse 10

2.1 Use Case 10

2.2 Maquettes 11

2.3 Diagramme d’activité / séquence 14

3 Concept 20

4.1 Diagramme de classe 20

4.2 Diagramme de séquence interactions 22

4.3 Diagramme d’entité-relation 24

4.4 Schéma relation de la base de données 25

5 Réalisation 28

5.1 Tests fonctionnels 28

5.2 Spécifications détaillées 28

5.3 Design du système 29

5.4 Configuration xyz 29

6 Test 30

6.1 Procédure de test 30

6.2 Protocol de test 30

6.3 Signature du protocole de test 30

7 Conclusion 31

7.1 Améliorations possibles 31

7.2 Auto-évaluation 31

8 Bibliographie: liste des sources et références 32

9 Glossaire 33

10 Signatures 34

11 Annexes 35

# Résumé du rapport du TPI

Le Résumé du rapport du TPI est une présentation conceptuelle du travail effectué et du résultat attendu d’au maximum une page A4 qui permettent au lecteur une rapide compréhension du rapport de travail. . Il contient trois paragraphes : Situation de départ, mise en œuvre, résultats.

Pour plus de détails, veuillez vous référez au Manuel ICT - partie B : Documentation / rapport du TPI - Question 14 - page 81.

# Les grandes lignes du projet

## Analyse de la situation initiale

Avant de commencer mon Analyse sur le projet, le projet n’existe pas. Les seules informations sur le projet se trouve dans le cahier de charge.

## Analyse de l’état désiré

A la fin de mon TPI le projet devra correspondre au cahier des charges, c’est-à-dire qu’un utilisateur pourras faire ces actions :

Si l’utilisateur est un joueur :

* Visualiser le classement et les matchs qui ont eu lieux.
* Ajouter un match contre un adversaire et y inscrire le score.
* Valider les matchs inscrits par les adversaires.

Si l’utilisateur est un admin :

* Inscrire des joueurs dans le tournois.

L’application aura donc un login qui permettra aux joueurs / admins de se connecter.

## Cahier des charges / exigences du système

### Analyse

* Analyser les tâches complètes du projet et les maquettes préparées auparavant
* Crée le diagramme des cas d’utilisation global.
* Créer les différents diagrammes de séquences des systèmes pour les applications.
* Créer les différents diagrammes d’activités principaux pour les applications

### Conception

* Créer le modèle relationnel de la base de données utile pour l’application Backend.
* Créer le diagramme des classes des applications.
* Créer les différents diagrammes d’interactions des méthodes principales des applications.

### Réalisation

* Réaliser l’implémentation de la base de données.
* Réaliser les différentes requêtes SQL utiles pour la base de données.
* Réaliser l’implémentation des services REST (application BackEnd).
* Développer l’interface homme-machine (application FrontEnd).
* Réaliser l’implémentation des applications et leur communication.
* Tester le fonctionnement complet des applications et leur relation.

## Organisation du projet

Mon TPI sera stocké sur un git ce qui permet d’accéder au projet entier à n’importe quel moment et n’importe où, ça me permet également de garder une trace des diverses versions de mon projet ce qui me permet de revenir en arrière à n’importe quel moment.

## Technologies

Pour réaliser mon TPI j’ai le choix entre deux technologies pour le client : Angular ou VueJS.

Pour le serveur j’avais le choix de savoir si j’utilisais JPA.

### Client

#### Angular

Angular est un framework robuste et puissant pour le développement d'applications web modernes. Il offre une architecture basée sur les composants, une liaison de données bidirectionnelle, une gestion avancée des routes, la gestion des dépendances, des outils de test et un écosystème solide.

#### VueJS

Vue.js est un framework JavaScript progressif, réactif et basé sur les composants. Il offre une approche flexible pour la construction d'interfaces utilisateur interactives, tout en étant léger, performant et bénéficiant d'un écosystème dynamique. C'est un choix populaire pour les développeurs qui cherchent une solution moderne et conviviale pour leurs projets web.

#### Choix de technologie

Pour mon TPI j’ai choisi le framework VueJS, voici les raisons pour lesquelles j’ai choisis VueJS plutôt qu’Angular :

1. VueJS est plus facile à apprendre que Angular, n’ayant aucune connaissance de base sur aucun des deux framework j’ai préféré choisir le framework sur lequel j’aurais le plus de facilité à apprendre.
2. VueJS est extrêmement flexible ce qui permet de choisir les fonctionnalités spécifiques en fonction du projet. La faite qu’il adopte une approche progressive est un plus car on adapter à n’importe quel moment en fonction des besoins.
3. VueJS est également beaucoup plus léger que Angular, ce qui permet à l’utilisateur une expérience plus rapide et où il y a le moins de chargement sur ordinateur comme sur téléphone.
4. Bien que la communauté d’Angular soit plus grande VueJS a également une communauté très active et évolutive avec de nombreux support et de ressources.

### Serveur

Pour le serveur je devais choisir si je voulais utiliser JPA pour communiqué avec ma base de données.

#### JPA

JPA, ou Java Persistence API, est une spécification Java qui simplifie la persistance des données dans les applications Java en fournissant une abstraction de la couche de persistance, un mapping objet-relationnel, une gestion des transactions, une portabilité et des fonctionnalités avancées.

#### Points forts de JPA

Je compte utiliser JPA pour mon TPI car il a des fonctionnalité assez intéressante qui me permettent de gagner du temps.

1. Fonctionnalités avancées : JPA offre également des fonctionnalités avancées telles que la gestion des relations entre les entités (relations One-to-One, One-to-Many, Many-to-One, Many-to-Many), les requêtes dynamiques, les mécanismes de mise en cache et l'héritage des entités.
2. Abstraction de la couche de persistance : JPA offre une abstraction de la couche de persistance, ce qui signifie que les développeurs n'ont pas à se soucier des détails spécifiques de la base de données sous-jacente. Ils peuvent travailler avec des objets Java familiers plutôt qu'avec des requêtes SQL complexes.
3. Gestion des transactions : JPA prend en charge la gestion des transactions, ce qui signifie qu'elle permet d'effectuer des opérations sur les objets persistants de manière cohérente et sécurisée. Elle offre des fonctionnalités telles que le contrôle des transactions, la gestion du commit et du rollback, assurant ainsi l'intégrité des données.

## Test technologique

### VueJs

#### Introduction

Vue.js est un framework JavaScript qui permet de créer des interfaces utilisateur interactives et réactives dans les applications web. Il offre une liaison de données bidirectionnelle entre le modèle de données et l'interface utilisateur, facilite la création de composants réutilisables et permet une intégration progressive dans les projets existants. Il est populaire pour sa facilité d'apprentissage, sa flexibilité et ses performances élevées.

#### Documentation

La documentation de VueJS est disponible ici : <https://vuejs.org/guide/introduction.html>

#### Création d’un projet VueJS

Pour crée un nouveau projet en vue il faut au préalable télécharger nodeJS afin de disposer de la commande « npm install » qui permet de télécharger des paquets de librairies.

Une fois que node est installé on peut utiliser la commande :

> npm init vue@latest

Cette commande installera et exécutera create-vue, l'outil officiel de création de projets Vue.Il faut ensuite répondre à des questions qui nous permettent de choisir d’installer certaine fonctionnalité de base :

✔ Project name: … <your-project-name>

✔ Add TypeScript? … No / Yes

✔ Add JSX Support? … No / Yes

✔ Add Vue Router for Single Page Application development? … No / Yes

✔ Add Pinia for state management? … No / Yes

✔ Add Vitest for Unit testing? … No / Yes

✔ Add Cypress for both Unit and End-to-End testing? … No / Yes

✔ Add ESLint for code quality? … No / Yes

✔ Add Prettier for code formatting? … No / Yes

Scaffolding project in ./< nouveau-projet >...

Done.

Il suffit ensuite de ce déplacé dans le projet qui viens d’être crée puis lancer le projet avec ces commandes.

> cd <nouveau-projet>

> npm install

> npm run dev

Une image contenant texte, capture d’écran, conception

Description générée automatiquementEt voilà l’application vue a été créé :

### SpringBoot

#### Introduction

Spring Boot est un framework open source pour le développement d'applications Java. Il simplifie le processus de création d'applications en fournissant une configuration et une infrastructure par défaut, permettant aux développeurs de se concentrer sur la logique métier. Il propose une approche "convention plutôt que configuration" en fournissant des valeurs par défaut intelligentes et en simplifiant la configuration. Spring Boot permet de créer rapidement des applications autonomes, des services web RESTful et des microservices, en offrant des fonctionnalités telles que la gestion des dépendances, la surveillance de l'application et la gestion de la sécurité. Il s'intègre étroitement avec l'écosystème Spring, ce qui en fait un choix populaire pour le développement d'applications Java robustes et évolutives.

#### Documentation

La documentation de SpringBoot est disponible ici : <https://spring.io/quickstart>

#### Création d’un projet SpringBoot

Pour crée un nouveau projet SpringBoot leur site propose un outil pour générer un nouveau projet : <https://start.spring.io/>.

Depuis ce site on peut donc configurer et ajouter des dépendances à notre projet, une fois terminé il suffit de générer et télécharger le zip. Le projet est généré automatique il n’y a rien à faire de plus.

#### Installer swagger

Swagger est un outil qui permet de documenter une API de manière efficace, il génère également une page web qui regroupe les diverses requêtes possibles à l’api.

Pour l’installer sur SpringBoot il suffit d’ajouter ces librairies au fichier pom.xml

<!-- Swagger UI Dependency -->

**<dependency>**

**<groupId>**org.springframework.boot**</groupId>**

**<artifactId>**spring-boot-starter-validation**</artifactId>**

**</dependency>**

**<dependency>**

**<groupId>**org.springdoc**</groupId>**

**<artifactId>**springdoc-openapi-starter-webmvc-ui**</artifactId>**

**<version>**2.0.2**</version>**

**</dependency>**

Il faut ensuite définir un Controller et crée une méthode qui correspondra a la requête de l’api.

@RestController

@CrossOrigin(origins = "\*")

@RequestMapping("api/v1/rencontre")

public class MailController {

  @Autowired

  private EmailSenderService senderService;

  @GetMapping(path = "/sendMail")

  public String triggerMail(String *mailTo*, String *subject*, String *body*) throws MessagingException {

    senderService.sendSimpleEmail(*mailTo*, *subject*, *body*);

    return "Mail bien envoyé";

  }

Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement}

Swagger détectera le Controller et on peut tester la requête depuis swagger après ça.

#### Envoyer des mails

Pour envoyer des mails avec Spring Boot la librairie JavaMail est recommandé. Il faut ajouter la dépendance dans le fichier pom.xml :

<!-- Java mail Dependency-->

**<dependency>**

**<groupId>**org.springframework.boot**</groupId>**

**<artifactId>**spring-boot-starter-mail**</artifactId>**

**</dependency>**

Une fois que la librairie est installer il faut configurer les propriétés du serveur d’envoie (SMTP) dans le fichier ‘application.properties’.

#Mail configuration

spring.mail.host=smtp.gmail.com

spring.mail.port=587

spring.mail.username=<mail>

spring.mail.password=<mot-de-passe>

spring.mail.properties.mail.smtp.auth=true

spring.mail.properties.mail.smtp.starttls.enable=true

Pour le test technologique j’ai utilisé les serveurs de Gmail, j’ai également utilisé le port 587 pour que ce soit le plus sécurisé possible. Il faut ensuite crée un service qui enverras les mails.

@Service

public class EmailSenderService {

  @Autowired

  private JavaMailSender mailSender;

  public String sendSimpleEmail(String *toEmail*, String *subject*, String *body*) {

    String result = "";

    try {

      MimeMessage mimeMessage = mailSender.createMimeMessage();

      MimeMessageHelper mimeMessageHelper = new MimeMessageHelper(mimeMessage, true);

      mimeMessageHelper.setFrom("killian.pasche7@gmail.com");

      mimeMessageHelper.setTo(*toEmail*);

      mimeMessageHelper.setSubject(*subject*);

      mimeMessageHelper.setText(*body*);

      mailSender.send(mimeMessage);

      result = "Mail envoyé !";

    } catch (MessagingException *e*) {

      result = "Erreur envoie de mail : " + e.getMessage();

    }

    return result;

  }

}

Pour tester la méthode j’ai créé une requête qui permet d’envoyer un mail depuis le swagger :

@RestController

@CrossOrigin(origins = "\*")

@RequestMapping("api/v1/mail")

public class MailController {

  @Autowired

  private EmailSenderService senderService;

  @GetMapping(path = "/sendMail")

  public String triggerMail(String *mailTo*, String *subject*, String *body*)

{

    return senderService.sendSimpleEmail(*mailTo*, *subject*, *body*);

  }

}

Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquementJ’ai ensuite lance l’application est entrée une adresse mail ainsi qu’un sujet et un message pour tester :

Une image contenant texte, capture d’écran, Police, Graphique

Description générée automatiquementComme on peut le voir ci-dessous j’ai bien reçu le mail :

# Analyse

## Use Case

Ce diagramme Use Case correspond à toutes les fonctionnalités principales de l’application FutNet Single Master. Chaque couleur représente une action différente.

Une image contenant diagramme, capture d’écran, texte, Caractère coloré

Description générée automatiquement

### Architecture

* Client : représente le coté front-end de l’application.
* HTTP/HTTPS : représente la liaison http/s entre le client et le serveur.
* Serveur : représente le coté back-end de l’application.

### Acteurs

#### Visiteur

Un utilisateur non-connecté qui arrive sur l’application.

* Se connecter : Permet à un visiteur de se connecter.

#### Joueur

Un utilisateur connecté qui a accès à l’application.

* Déconnexion : Permet à un joueur de se déconnecter.
* Visualiser classement / rencontre : Récupère le classement et les matchs pour les affichés.
* Ajouter une rencontre : Permet à un joueur d’ajouter une rencontre.
* Modifier une rencontre : Permet à un joueur de modifier une rencontre qui lui appartient.
* Valider une rencontre : Permet à un joueur de modifier l’état d’une rencontre qui ne lui appartient pas.
* Supprimer une rencontre : Permet à un joueur de supprimer une rencontre qui lui appartient s’il n’est pas validé.

#### Admin

Un utilisateur connecté qui a accès à l’application admin. L’utilisateur admin inclus tous ce qu’un joueur peut faire.

* Supprimer un joueur : Permet à l’admin de supprimer un joueur s’il n’a joué aucun match.
* Modifier un joueur : Permet à l’admin de modifier les informations d’un joueur.
* Inscrire un joueur : Permet à l’admin d’inscrire un joueur.

#### Base de données

La base de données s’occupe de gérer les données qu’on lui envoie.

## Maquettes

Voici les maquettes des différentes pages qu’on pourra voir sur l’application web :

### Login

Une image contenant texte, capture d’écran, conception

Description générée automatiquementEcran d’accueil pour les visiteurs, c’est via cette écran qu’ils pourront se connecter.

### Classement

Sur cette page les joueurs pourront voir le classement générale des utilisateurs.

Une image contenant texte, capture d’écran, nombre, logiciel

Description générée automatiquement

### Rencontre

C’est sur cette page que les joueurs pourront voir les matchs, en ajouter, en modifier et en valider.

### Une image contenant texte, capture d’écran, logiciel, nombre Description générée automatiquementInscription

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquementC’est dans cette page que les admins pourront ajouter, modifier, supprimer des utilisateurs.

## Diagramme d’activité / séquence

### Une image contenant texte, diagramme, capture d’écran, Plan Description générée automatiquementUne image contenant texte, diagramme, Parallèle, capture d’écran Description générée automatiquementInscription de joueur

Pour inscrire un joueur, l’admin devra remplir un formulaire, une fois remplis la partie client vas s’occuper d’envoyer les données au serveur si toutes les informations sont remplies. Si les informations ne sont pas remplies un message d’erreur va s’afficher et la page sera rafraîchis. Une fois que le serveur a reçu les données il va s’occupé de les contrôlées, si elles sont valables il va demander à la base de données si un utilisateur avec ces données existe, si non un mot de passe ainsi qu’un nom d’utilisateur sera généré. Une fois la génération terminé l’utilisateur sera enregistré dans la base de données. Si l’inscription est réussite un mail sera envoyé à l’utilisateur avec le mot de passe et le nom d’utilisateur. Le serveur ensuite retourne l’état de la requête pour que le client affiche le message en conséquence, c’est-a-dire que si les données ne sont pas valables, l’utilisateur existe ou que l’inscription à échoué l’application retournera KO et affichera un message d’erreur. En cas contraire l’application retournera OK et affichera un message de succès.

### Une image contenant texte, capture d’écran, diagramme, Parallèle Description générée automatiquementUne image contenant texte, diagramme, capture d’écran, ligne Description générée automatiquementSe connecter

Lorsqu’un visiteur voudra se connecter sur l’application il devra entrer son login et le client va vérifier les données, elles sont null un message d’erreur sera afficher, sinon les données seront transmises au serveur, le serveur va s’occuper d’appeler la Base de donnée pour savoir si un utilisateur existe avec ce login, si oui il va créer une session et retourner ok, le client va ensuite afficher la page en mode « utilisateur enregistré ». Si non il va retourner KO et afficher un message d’erreur sur le client.

### Une image contenant texte, diagramme, ligne, Rectangle Description générée automatiquementValider une rencontre

Une image contenant texte, diagramme, capture d’écran, Parallèle

Description générée automatiquementLorsqu’un joueur validera un match le client enverras la rencontre au serveur, le serveur s’occupera de contrôler que la rencontre appartient bien au joueur, si elle appartient bien au joueur la base de données modifiera la rencontre. Si la rencontre a bien été modifier le serveur retournera OK et le client affichera un message de succès. Si la rencontre n’a pas été modifié ou que la rencontre appartient à un autre joueur il retournera KO et le client affichera un message d’erreur.

### Une image contenant texte, capture d’écran, diagramme, Parallèle Description générée automatiquementSupprimer une rencontre

Une image contenant texte, capture d’écran, diagramme, ligne

Description générée automatiquement

Lorsqu’un joueur supprimera une rencontre, la rencontre sera envoyée au serveur SpringBoot, le serveur vérifiera si la rencontre a déjà été validé, il vérifie également si la rencontre envoyée appartient au joueur qui l’as envoyé et qu’il se le joueur 1 de cette rencontre. Si ces deux tests sont passé la rencontre sera alors supprimer de la base de données. Si la rencontre à bien été supprimé un message de succès sera retourner. En cas contraire, si la rencontre est valide, qu’elle n’appartient pas au joueur 1 ou qu’elle n’a pas été supprimer, un message d’erreur sera alors retourné.

### Une image contenant texte, diagramme, Parallèle, capture d’écran Description générée automatiquement Modifier un match

Une image contenant texte, ligne, diagramme, Tracé

Description générée automatiquement

Lorsqu’un joueur aura modifié une rencontre le client l’enverra au serveur, le serveur regardera si la rencontre est validée et il vérifie également si la rencontre envoyée appartient au joueur qui l’as envoyé et qu’il se le joueur 1 de cette rencontre. Si les tests sont réussis la Base de données modifiera la rencontre. Le serveur vérifie si la rencontre à été bien modifié, si c’est le cas il retourne un message de succès. Si en cas contraire la rencontre n’a pas été modifier ou bien la rencontre était valide ou elle n’appartenait pas au joueur 1, le serveur retournera un message d’erreur. Le client s’occupe d’afficher le message retourné pas le serveur.

# Concept

# Conseil: D’autres chapitres peuvent s’ajouter à cette partie en fonction du projet.

## Diagramme de classe

### Partie client

Une image contenant texte, diagramme, capture d’écran, Rectangle

Description générée automatiquementVoici mon diagramme de classe de la partie cliente, le diagramme représente la hiérarchie de mon application en vue js :

index.html : C’est le point de départ de mon application, c’est dans ce fichier que VueJS va écrire son code.

main.js : c’est le fichier JavaScript qui initialisera un composant dans la page index.html. Il est également responsable de la configuration des plugins et de composants tiers qui seront utilisés.

App.vue : c’est la racine de l’application définie dans Vue.js.

Login.vue/Index.vue/Classement.vue/Rencontre.vue/Inscription.vue : Ce sont les différentes vues que l’application possède.

« component » : les éléments possédant cette balise sont les composants diverse à mon application.

router.js axios.js : ce sont des librairies de vue qui sont utilisée dans le projet, router est utilisé pour faire une single app page et axios est utilisé pour faire des requêtes sur le serveurs SpringBoot

### Partie serveur

Une image contenant texte, diagramme, capture d’écran, Plan

Description générée automatiquementVoici le diagramme de classes de la partie serveur. Il est décomposé en 4 parties :

* Verte : cette partie contient les controllers, c’est le point d’entrée des requêtes.
* Orange : cette partie contient les services. Ce sont les services qui font la relation entre le ctrl et la DB. On peut comparer les services à des WRK.
* Bleu Claire : cette partie contient les dépôts (Repository). Les dépôts s’occupent de parler avec la DB.
* Bleu : Cette partie corresponds aux entités, chaque entité représente une table de la DB.

## Diagramme de séquence interactions

### Inscription

Une image contenant texte, capture d’écran, diagramme, Parallèle

Description générée automatiquementCe diagramme représente le déroulement de ce qui passe lors de l’inscription d’un utilisateur.

1. Le client envoie une requête au serveur SpringBoot avec un utilisateur en paramètre
2. Le serveur contrôle que l’utilisateur ne soit pas null.
   1. S’il est null il retournera une error 400
3. Le serveur va ensuite récupérer toutes les informations de l’utilisateur dans son repository (role, équipe, tournoi)
4. Il va ensuite générer un mot de passe ainsi qu’un username.
5. L’utilisateur sera enregistré dans la base de données.
6. Un mail sera envoyé à l’utilisateur concerné pour lui communiquer ces informations de login.
7. Le serveur retournera un string si tous c’est bien passé.

### Validation d’une rencontre

Une image contenant texte, diagramme, capture d’écran, Parallèle

Description générée automatiquementCe diagramme représente ce qu’il se passe lorsqu’un utilisateur souhaite valider une rencontre.

1. Le client envoie une requête au serveur SpringBoot avec l’identifiant de la rencontre en paramètre
2. Le Controller appelle le service qui s’occupe des rencontres.
3. Le Service controlle que la session est existante
   1. Si non : Renvoie un message d’erreur au client
4. Le service récupère la rencontre qui a été envoyée
   1. Si la rencontre n’existe pas : renvoie un message d’erreur au client
5. Le service récupère l’entrée qui correspond à l’utilisateur connecté et la rencontre.
   1. Si l’entrée n’est pas joueur 1 : Renvoie un message d’erreur au client
6. Modifie la rencontre pour la faire passer à valide
7. Renvoie un message de succès au client.

## Diagramme d’entité-relation

Une image contenant diagramme, Rectangle, ligne, capture d’écran

Description générée automatiquementLe diagramme d’entité-relation représente les différentes relations que les entités on envers les autres. Dans le diagramme il y’a trois relation de type 0 à N (0…\*). Ces trois relations partent toute de la table Utilisateur, vers les tables (Tournois, Rôle, Equipe) :

Un utilisateur participe à un tournoi.

Un utilisateur appartient à un role.

Un utilisateur joue dans une équipe.

Il y’a une relation de plus qui part de la table utilisateur, cette relation est un peu spéciale, puisque c’est une relation N à N (0…\* a 0…\*). Cela signifie que lorsque la base de donnée sera créé il faudra une table de relation entre utilisateur et rencontre.

Un utilisateur joue 0 ou plusieurs rencontres. Une rencontre peut être jouée par 0 ou plusieurs joueurs.

## Schéma relation de la base de données

Pour la nomenclature de ma base de données j’ai choisis une nomenclature assez standard, qui est souvent utilisé.

### Tournoi

La table tournoi est une table un peu spécial vue qu’elle n’est pas nécessaire au projet, cette table est en prévision pour un futur ou l’admin pour crée plusieurs tournois et choisir quel tournoi est actifs. Pour le moment cette table n’est pas vraiment nécessaire, elle est surtout utilisée pour que des utilisateurs soit attribué à un tournois. Voilà ce qu’elle contient :

* id\_tournoi : l’identifiant unique du tournois
* nom : le nom du tournoi

### Utilisateur

La table utilisateur contient tous les joueurs / admin du tournois, Voilà ce qu’elle contient :

* id\_utilisateur : l’identifiant unique d’un utilisateur.
* nom : le nom de l’utilisateur
* prenom : le prenom de l’utilisateur
* mail : l’adresse mail de l’utilisateur
* username : le nom d’utilisateur de l’utilisateur
* password : le mot de passe de l’utilisateur
* fk\_role : référence à la table « role », contient le rôle de l’utilisateur
* fk\_equipe : référence à la table « équipe », contient l’équipe de l’utilisateur.
* fk\_tournoi : référence à la table « tournoi », contient le tournoi dans lequelle l’utilisateur est inscrit.

### Role

La table role contient les différents rôles qu’un utilisateur peut se voir attribué. Voilà ce qu’elle contient :

* pk\_role : l’identifiant unique d’un rôle
* role : le rôle.

### Equipe

La table equipe contient les différentes équipes qu’un utilisateur peut se voir attribuer. Voilà ce qu’elle contient :

* pk\_equipe : L’identifiant unique d’une équipe
* nom\_equipe : le nom de l’équipe

### Rencontre

La table rencontre définis les matchs que les joueurs peuvent faire entre eux. Cette table ne contient pas tous les attributs d’une rencontre, elle est compléter par la table de relation utilisateur-rencontre. Voilà ce qu’elle contient :

* pk\_rencontre : l’identifiant unique de la rencontre.
* valide : si la rencontre a été validée.
* date : la date de la rencontre

### Utilisateur\_rencontre

La table utilisateur\_rencontre est une table de relation, c’est-à-dire qu’elle contient les relations entre utilisateur et rencontre. On pourrait supprimer cette table et définir deux utilisateurs ainsi que deux scores dans la table rencontre, mais je pense que c’est mieux de cette manière, par exemple lorsque l’on devras récupéré le score de chaque joueur, on aurait dû récupéré le nom de la colonne pour savoir si il est joueur 1 ou joueur 2. Le faite qu’une table de relations soit entre utilisateur et rencontre simplifie la récolte de résultat entre les deux tables.

Voilà ce qu’elle contient :

* id\_utilisateur : Référence à un utilisateur.
* id\_rencontre : Référence à une rencontre.
* score : Score du joueur.
* est\_joueur1 : définis le joueur 1 d’une rencontre.

« est\_joueur1 » permet de définir le joueur qui aura le droit de modifier ou valider une rencontre. Cela permettra aussi de définir le joueur qui aura le score 1 ou le score 2.

# Réalisation

## Tests fonctionnels

Tous les tests fonctionnels seront réalisés sur postman

### Login

|  |  |  |
| --- | --- | --- |
| **Test :** | **Résultat attendue :** | **Résultat obtenue :** |

### Inscription

|  |  |  |
| --- | --- | --- |
| **Test :**  Inscrire un joueur | **Résultat attendue :**  L’utilisateur crée / un mail reçu | **Résultat obtenue :** |

### Modifier une rencontre

|  |  |  |
| --- | --- | --- |
| **Test :** | **Résultat attendue :** | **Résultat obtenue :** |

### Modifier un utilisateur

|  |  |  |
| --- | --- | --- |
| **Test :** | **Résultat attendue :** | **Résultat obtenue :** |

### Valider une rencontre

### Supprimer une rencontre

### Supprimer un utilisateur

## Implémentation de la base de données

Lors de l’implémentation de la base de données, je me suis rendu compte que le schéma actuel va poser un problème dans le futur. C’est-à-dire qu’un utilisateur pouvait seulement participer un tournois. Et c’est un problème dans le sens que on ne veut pas inscrire les même joueur chaque année. C’est pour cela que j’ai supprimé cette table de mon schéma.

### Une image contenant texte, capture d’écran, Police, nombre Description générée automatiquementSchéma actuelle

Voilà le schéma utiliser actuellement dans la base de données. On peut voir que la table tournoi à disparu. Il n’y a donc pas de gestion de tournoi dans le projet.

Un nouveau champ dans la table rôle est également apparut. Il permet de gérer le niveau du rôle.

### Schéma futur

Dans un futur ou on aurait besoin de gérer les tournois j’ai également réalisé ce schéma qui permet de voir à ce que peut ressembler la base de données :

Une image contenant texte, capture d’écran, diagramme, Police

Description générée automatiquementIl y a dans ce schéma une table de relation entre utilisateur et tournoi. Cela permet de définir plusieurs tournois à un utilisateur. Une relation entre rencontre et tournoi a aussi été ajoutée. Cela permet de définir à quel tournoi appartient les rencontre vue qu’un utilisateur peut avoir plusieurs tournois.

## Réalisation du Serveur

### Gestion de session

### Différentes requêtes

#### Login

#### Logout

#### getAllUsers

#### saveUser

#### putUser

#### deleteUserById

#### getClassement

#### getRencontres

#### saveRencontre

#### putValideRencontre

#### putRencontre

#### deleteRencontreById

## Configuration xyz

Conseil: D’autres chapitres peuvent s’ajouter à cette partie en fonction du projet.

# Test

## Procédure de test

Représentation du processus de test..Qui teste quoi à l’aide de quels moyens ?

## Protocol de test

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nr. | Objet testé | Description du test | Attente | Résultat | Visa |
| 1.0 |  |  |  |  |  |

Protocole de test détaillé avec Visa.

## Signature du protocole de test

|  |  |  |
| --- | --- | --- |
| Date | Nom | Signature |
|  |  |  |

# Conclusion

Conclusion sur le projet en général. Les objectifs sont-ils atteints ? Si non, pourquoi ?

## Améliorations possibles

Y a-t-il des améliorations possibles (nouvelles fonctionnalités, amélioration et/ou modification du design, modification du support de données…) ?

## Auto-évaluation

Quels sont mes sentiments vis-à-vis du travail effectué. Enoncé des facteurs qui expliquent le succès ou l’échec de la réalisation des objectifs du projet.

# Bibliographie: liste des sources et références

Conseil: Cette partie peut contenir une liste des sources et éléments littéraires utilisés.

# Glossaire trier par ordre alphabetique

|  |  |
| --- | --- |
| Terme | Signification |
|  |  |
|  |  |
|  |  |

# Signatures

Je soussigné déclare que les informations contenues dans ce rapport de travail pratique individuel rendu ce jour le XX.XX.20XX dans le cadre de la procédure de qualification de mon CFC d’informaticien/-ne, ne sont pas plagiées. Toutes les informations de sources extérieures ainsi que les informations fournies par des tiers durant le déroulement du travail sont consignées.

L’apprenti/-e doit signer la documentation avant la remise afin de témoigner de la justesse des informations qui y figurent.

|  |  |  |
| --- | --- | --- |
| Date | Nom | Signature |
|  |  |  |

# Annexes

Code imprimé, protocole de séances, etc.: spécifier complètement et annexer à la documentation.

Listings de scripts et programmes. Le travail fourni par la candidate ou le candidat doit être entièrement documenté. Omettre le code généré automatiquement s’il n’est pas nécessaire à la compréhension.

Dans le cas où des manuels ont été créés, vous pouvez les ajouter ici en tant qu’annexe.