PROJET - API de Gestion de Tournois E-sport

Formation : Node.js - Master Européen Expert IT - Hesias
Modalités : Travail individuel

CONSIGNES GÉNÉRALES

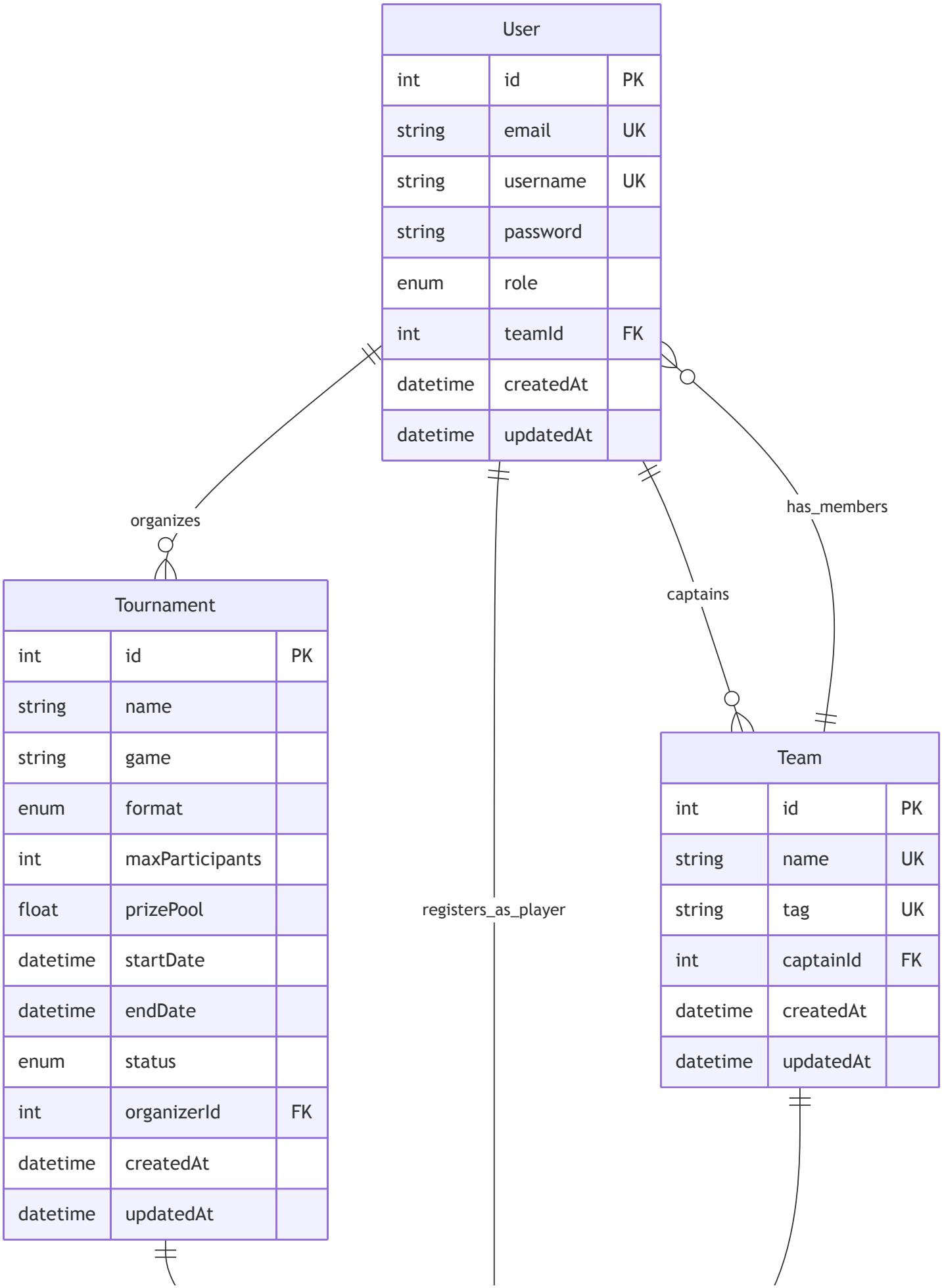
Ce travail pratique évalue votre capacité à concevoir et développer une API REST complète avec Node.js et Express. Vous devez développer une application de gestion de tournois e-sport en respectant les contraintes architecturales et fonctionnelles décrites dans ce document.

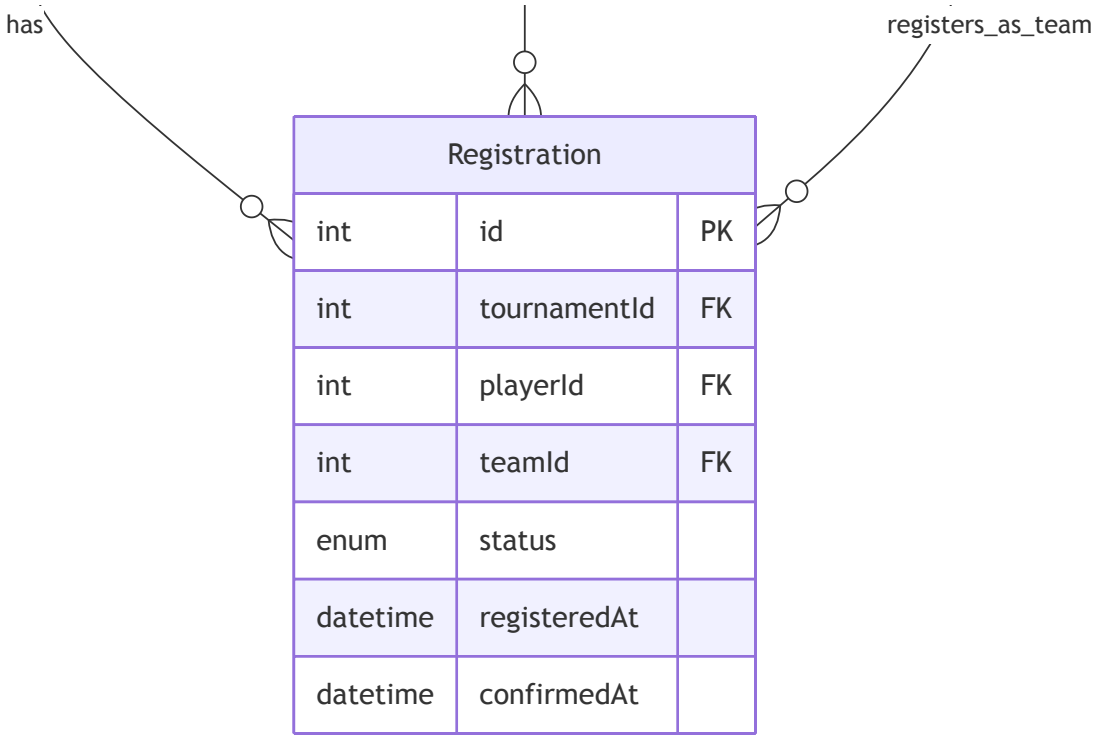
CONTEXTE

Vous êtes chargé de développer le backend d'une plateforme de gestion de tournois e-sport. Cette API permettra de gérer des tournois, des équipes et des inscriptions avec différents niveaux de permissions.

MODÈLE DE DONNÉES

Diagramme de classes


User
int id PK
string email UK
string username UK
string password
enum role
int teamId FK
datetime createdAt
datetime updatedAt
organizes
captains
has_members
registers_as_player
Tournament
int id PK
string name
string game
enum format
int maxParticipants
float prizePool
datetime startDate
datetime endDate
enum status
int organizerId FK
datetime createdAt
datetime updatedAt
Team
int id PK
string name UK
string tag UK
int captainId FK
datetime createdAt
datetime updatedAt



Légende :

- PK : Clé primaire
- FK : Clé étrangère
- UK : Contrainte d'unicité
- ||--o{ : Relation un-à-plusieurs (1-N)
- ||--o| : Relation un-à-un (1-1)

Enums

UserRole : 'PLAYER', 'ORGANIZER', 'ADMIN'

TournamentFormat : 'SOLO', 'TEAM'

TournamentStatus : 'DRAFT', 'OPEN', 'ONGOING', 'COMPLETED', 'CANCELLED'

RegistrationStatus : 'PENDING', 'CONFIRMED', 'REJECTED', 'WITHDRAWN'

FONCTIONNALITÉS ATTENDUES

A. Authentification et Autorisation (3 points)

A.1 Routes et validations (1.5 point)

- `POST /api/auth/register` : Créer un utilisateur et retourner un JWT
- `POST /api/auth/login` : Vérifier credentials et retourner un JWT
- **Validations** :
  - Email : format valide et unique en base
  - Username : 3 à 20 caractères (alphanumérique + underscore), unique
  - Password : minimum 8 caractères avec au moins 1 majuscule, 1 minuscule, 1 chiffre
  - Role : par défaut 'PLAYER'
- **Sécurité** : Hasher le password avec bcrypt, générer JWT avec expiration (ex: 24h)

A.2 Middlewares de sécurité (1.5 point)

- `authenticate` : Vérifier le token JWT et attacher req.user
- `authorize(roles)` : Vérifier que req.user.role est autorisé (retour 403 si refusé)
- Application correcte du RBAC sur toutes les routes protégées

B. Gestion des Tournois (4 points)

B.1 Routes CRUD (1.5 point)

- `GET /api/tournaments` : Liste avec filtres (?status, ?game, ?format) et pagination (?page, ?limit)
- `GET /api/tournaments/:id` : Détails d'un tournoi
- `POST /api/tournaments` : Créer (ORGANIZER ou ADMIN uniquement)
- `PUT /api/tournaments/:id` : Modifier (ORGANIZER ou ADMIN uniquement)
- `DELETE /api/tournaments/:id` : Supprimer (ORGANIZER ou ADMIN uniquement)

B.2 Validations métier (1.5 point)

- **Cohérence format/inscriptions** : Un tournoi SOLO n'accepte que des joueurs individuels, un tournoi TEAM n'accepte que des équipes
- **Dates** : À la création, startDate doit être future. Si endDate est renseignée, elle doit être après startDate
- **Statut initial** : Toujours créé en DRAFT
- **Protection modification** : Un tournoi COMPLETED ou CANCELLED ne peut plus être modifié
- **Protection suppression** : Impossible de supprimer un tournoi ayant des inscriptions CONFIRMED

B.3 Gestion des statuts (1 point)

- Route : `PATCH /api/tournaments/:id/status` avec { status: "OPEN" }

- **Transitions autorisées** :
  - DRAFT → OPEN : Vérifier que startDate > maintenant
  - OPEN → ONGOING : Vérifier qu'il y a au moins 2 participants CONFIRMED
  - ONGOING → COMPLETED : ADMIN uniquement
  - N'importe quel statut → CANCELLED : Créateur ou ADMIN

C. Gestion des Équipes (2 points)

C.1 Routes CRUD (1 point)

- `GET /api/teams` : Liste toutes les équipes
- `GET /api/teams/:id` : Détails avec captain (include relation)
- `POST /api/teams` : Créer (utilisateur authentifié devient capitaine)
- `PUT /api/teams/:id` : Modifier (capitaine uniquement)
- `DELETE /api/teams/:id` : Supprimer (capitaine uniquement)

C.2 Validations (1 point)

- **Nom** : 3 à 50 caractères, doit être unique
- **Tag** : 3 à 5 caractères (majuscules et chiffres uniquement), doit être unique
- **Rôle capitaine** : Vérifier que le capitaine a role = 'PLAYER'
- **Protection suppression** : Impossible de supprimer si l'équipe est inscrite à un tournoi actif (status OPEN ou ONGOING)

D. Inscriptions aux Tournois (4 points)

D.1 Routes (1 point)

- `POST /api/tournaments/:tournamentId/register` : Inscription SOLO ou TEAM
- `GET /api/tournaments/:tournamentId/registrations` : Liste des inscriptions
- `PATCH /api/tournaments/:tournamentId/registrations/:id` : Modifier statut (organisateur ou participant)
- `DELETE /api/tournaments/:tournamentId/registrations/:id` : Annuler si PENDING

D.2 Validations métier (3 points)

- **Statut tournoi** : Inscription possible uniquement si tournoi en status OPEN

- **Cohérence format** : Un tournoi SOLO n'accepte que des inscriptions de joueurs individuels (playerId renseigné, teamId null), un tournoi TEAM n'accepte que des inscriptions d'équipes (teamId renseigné, playerId null). Une inscription ne peut avoir les deux simultanément.
- **Limite participants** : Compter les inscriptions CONFIRMED et refuser si maxParticipants atteint
- **Unicité** : Un joueur ou une équipe ne peut s'inscrire qu'une seule fois au même tournoi
- **Capitaine équipe** : Pour inscription TEAM, vérifier que l'utilisateur qui inscrit l'équipe en est bien le capitaine
- **Annulation** : Une inscription CONFIRMED ne peut être supprimée (DELETE), mais peut passer en WITHDRAWN via PATCH

E. Architecture et Structure (3 points)

E.1 Architecture MVC (1.5 point)

- Séparation stricte routes / controllers / services (pas de logique métier dans routes)
- Schéma Prisma complet avec toutes les relations
- Migrations appliquées et fichier seed fonctionnel

E.2 Middlewares et configuration (1.5 point)

- Middleware errorHandler centralisé (4 paramètres, placé en dernier)
- asyncHandler ou gestion appropriée des erreurs async
- Validation Zod sur toutes les routes POST/PUT/PATCH
- Configuration .env (.env.example présent, .env ignoré)
- README complet (installation, configuration, scripts)

F. Qualité du Code (4 points)

F.1 Lisibilité et maintenabilité (2 points)

- Nommage cohérent et explicite (variables, fonctions, fichiers)
- Respect du principe DRY (éviter le code dupliqué, helpers/utils)
- Code formaté de manière cohérente
- Structure de fichiers logique et claire
- Commentaires pertinents uniquement sur logique complexe

F.2 Gestion des erreurs et sécurité (2 points)

- Gestion complète des erreurs avec messages clairs
- Status codes HTTP appropriés (400, 401, 403, 404, 500)
- Aucun secret en dur (JWT_SECRET, DATABASE_URL en .env uniquement)
- Pas de console.log de debug oubliés
- Tests manuels effectués et fonctionnalités validées

FONCTIONNALITÉS BONUS (5 points maximum)

Bonus 1 : Documentation Swagger (1 point)

- Installation de swagger-jsdoc et swagger-ui-express
- Documentation complète de toutes les routes avec JSDoc
- Interface accessible sur /api-docs
- Schémas réutilisables dans components/schemas
- Exemples de requêtes et réponses

Bonus 2 : Tests Vitest (2 points)

- Minimum 15 tests (unitaires et intégration)
- Tests des services (logique métier)
- Tests des routes API avec Supertest
- Au moins 3 cas d'erreur testés (400, 401, 403, 404)
- Tests qui passent tous

Bonus 3 : Statistiques Tournoi (1 point)

- Route `GET /api/tournaments/:id/stats`
- Retourne :
  - Nombre total d'inscriptions
  - Répartition des inscriptions par statut (PENDING, CONFIRMED, REJECTED, WITHDRAWN)
  - Pourcentage de remplissage (inscriptions CONFIRMED / maxParticipants)
  - Liste des participants confirmés avec leurs informations

Bonus 4 : Interface Web EJS (1 point)

- Page d'accueil : liste des tournois ouverts
- Page détail tournoi : informations et inscriptions

- Page inscription : formulaire
- Utilisation de Tailwind CSS

RÉPARTITION DES POINTS

Catégorie	Points
A. Authentification et Autorisation	3
B. Gestion des Tournois	4
C. Gestion des Équipes	2
D. Inscriptions aux Tournois	4
E. Architecture et Structure	3
F. Qualité du Code	4
Total	**20**
Bonus (maximum)	+5

LIVRABLES

Repository Git

- Historique de commits régulier
- Messages de commit descriptifs (format : feat:, fix:, refactor:)
- .gitignore correct (.env, node_modules, dist, coverage)

Fichiers obligatoires

- README.md : installation, configuration, scripts, documentation API
- .env.example : template des variables d'environnement
- package.json : dépendances et scripts
- Schéma Prisma avec migrations

Contenu du README.md

Le fichier README.md doit contenir au minimum :

- Titre du projet
- Instructions d'installation
- Configuration requise (.env)
- Commandes pour initialiser la base de données
- Commande pour démarrer le serveur
- Documentation de l'API ou lien vers Swagger (bonus)

CRITÈRES D'ÉVALUATION

Fonctionnalités (13 points)

L'ensemble des fonctionnalités obligatoires doit être implémenté et fonctionnel. Les validations métier sont particulièrement importantes et seront vérifiées en détail.

Architecture (3 points)

Le respect strict de l'architecture MVC est attendu. La séparation des responsabilités entre routes, controllers et services doit être claire. L'utilisation appropriée de Prisma et des middlewares est évaluée.

Qualité du code (4 points)

Le code doit être lisible, maintenable et respecter les bonnes pratiques. La gestion des erreurs doit être complète et appropriée. Aucun secret ne doit être présent dans le code source.

Pénalités applicables

- Secrets commités dans le repository : -1 point
- Passwords non hashés : -1 point
- Absence de validation des entrées utilisateur : -1 point
- Fichier .env non ignoré : -0.5 point

MODALITÉS DE RENDU

Format : Lien vers repository Git (GitHub, GitLab ou autre)
Accès : Le repository doit être accessible au moment de l'évaluation

Le repository doit contenir l'ensemble du code source ainsi que les fichiers de configuration nécessaires au démarrage du projet (.env.example, README.md, etc.).

RESSOURCES AUTORISÉES

- Documentation officielle (Node.js, Express, Prisma, Zod, JWT, bcrypt, Vitest)
- Supports de cours
- Notes de cours personnelles
- Projet fil rouge (LAN Party Manager)
- Recherches sur la documentation technique

Non autorisé : Copie de solutions complètes, partage de code entre étudiants.