

**Département Informatique**

 71 rue Peter Fink

01000 Bourg-en-Bresse

 04 74 45 50 59

*Gaël Journet Jean-François Marcourt Nathan Ozimek*

Promotion 2022/2023

Tuniv

rapport de projet

Adrien Peytavie

Lionel Buathier

Diplôme Universitaire de Technologie

Informatique

|  |
| --- |
| [Nom de la société] |
|  |

Parcours IEM :   
Informatique Et Mobilité

Remerciements

Nous adressons des remerciements à Lionel Buathier et Adrien Peytavie pour leur encadrement tout au long du projet, ainsi qu’à Émilien Nicolas et Benjamin Chazelle pour leur aide technique.

Table des matières

[I. Introduction 1](#_Toc129951742)

[I.1 Objectifs et contexte du projet 1](#_Toc129951743)

[I.2 Présentation de l’Equipe 1](#_Toc129951744)

[II. Réalisation du Projet 1](#_Toc129951745)

[II.1 Présentation général du projet 1](#_Toc129951746)

[II.2 Architecture 1](#_Toc129951747)

[II.3 Nom du module 1, … 2](#_Toc129951748)

[III. Resultats 2](#_Toc129951749)

[IV. Conclusion 2](#_Toc129951750)

[V. Références 3](#_Toc129951751)

# Introduction

## Objectifs et contexte du projet

Le projet faisait suite à notre projet de SAÉ du premier semestre, Tuniv, qui visait à développer un site web de gestion de tournois de sport universitaire où des administrateurs pouvaient créer des tournois et y assigner des équipes pour que les arbres de tournois soient ensuite générés automatiquement. Des arbitres étaient assignés aux différents matchs et pouvaient entrer et mettre à jour les résultats en temps réel. Chaque match et chaque tournoi restait accessible via une base de données stockant tous les résultats des matchs actuels comme passés.

Notre projet du second semestre visait quant à lui à améliorer le projet que nous avions produit à l’issue du premier semestre, en se focalisant particulièrement sur la sécurité, l’ergonomie, la qualité logicielle et l’accessibilité, le tout en mettant en pratique les nouveaux savoirs et compétences acquis au cours de ce deuxième semestre.

Ce rapport détaillera la réalisation de ce projet, en présentant son architecture, les différents modules mis en place pour atteindre les objectifs d’amélioration fixés, ainsi que le résultat final, sans omettre de mentionner les améliorations possibles et les fonctionnalités prévues mais finalement non développées.

## Présentation de l’Equipe

L’équipe est composée de Gaël Journet, Jean-François Marcourt et Nathan Ozimek.

Gaël Journet

Jean-François Marcourt a travaillé en tant que Project-Leader et fut principalement en charge de l’ensemble des éléments visuels autour du projet (UI/UX/Présentations), ainsi que des tâches liées à l’optimisation de la sécurité logiciel.

Nathan Ozimek a travaillé en tant que développeur Backend principalement en charge de l’amélioration de la sécurité et de l’ergonomie.

Malgré des rôles bien définis, l’organisation mise en place autour du projet a permis à chacun des membres du groupe de contribuer à l’ensemble des tâches effectuées.

# Réalisation du Projet

## Présentation générale du projet

Les améliorations apportées tout au long du projet relèvent de quatre axes principaux : la sécurité, l’ergonomie, la qualité logicielle et l’accessibilité.

En termes de sécurité, le changement le plus impactant a été le passage d’une architecture de site web « classique » à une architecture routeur, comme vu dans le module d’architecture logicielle : le serveur est désormais lancé sur une page spécifique qui contient des routes définies manuellement vers chaque page. D’autres changements ont été mis en place, comme le chiffrement des mots de passe utilisateurs dans la base de données et des protections contre des attaques classiques comme les failles XSS ou le forçage des mots de passe, à l’aide notamment d’un système de captcha.

[schéma routeur, screen chiffrement]

En termes d’ergonomie, des changements ont été mis en place au niveau de la gestion des tournois, avec l’automatisation de l’avancement des tournois lorsque tous les matchs existants étaient clôturés, la possibilité pour un administrateur de modifier le score d’un match même après qu’il ait été clôturé ainsi qu’une révision graphique modeste.

[screen automatisation]

En termes de qualité logicielle, le passage à l’architecture routeur mentionné précédemment a permis de centraliser certaines parties du code dans la page de routage, notamment la connexion à la base de données ; un repassage a également été effectué sur tous les fichiers afin de retirer les fichiers devenus inutiles, de passer du CSS au SCSS pour l’affichage et de faire un nettoyage complet du code afin de le rendre plus lisible et moins brouillon.

[screen comparaison avant après routeur et commentaires]

Enfin, en termes d’accessibilité, des pages de contact et de support ont été mises en place pour permettre aux utilisateurs de contacter les administrateurs afin de faire des retours ou des requêtes.

[screen page contact/support]

## Architecture

Le site est structuré dans différentes pages web PHP, divisée en plusieurs dossiers :

* Le dossier ‘webroot’, qui contient la page de départ index et un sous-dossier « assets » dans lesquels se trouvent tous les fichiers nécessaires à l’affichage de chaque page (images, feuilles de style SCSS). La page index, quant à elle, renvoie directement à une page app, située à la racine du projet.
* La page app est le centre de l’architecture routeur du projet, car elle permet d’accéder à tous les autres fichiers. C’est également ici qu’est initialisée la connexion à notre base de données MariaDB, qui utilise la classe PDO de PHP (comme cette page renvoie à toutes les autres, elles ont donc toutes accès à la base de données).
* Les vues des autres pages sont stockées dans un dossier page et accessibles grâce aux routes mises en place dans le fichier routeur app. Quant aux traitements des informations liées aux formulaires et autres entrées utilisateurs, ils sont effectués dans des fichiers de configuration, stockés dans un dossier à part nommé config. Lorsqu’un traitement est requis, c’est le routeur qui renvoie vers la page de configuration correspondante, qui traite les données et modifie la base de données si besoin avant de renvoyer l’utilisateur vers la page d’origine (ou une page correspondante dans le cas de la création d’un nouveau tournoi par exemple).

Le choix a été fait de ne pas utiliser de framework web comme CakePHP car compte tenu de la taille du projet au début du deuxième semestre, refaire toute l’architecture en utilisant un framework nous aurait coûté plus de temps que nous n’en aurions gagné.

La sécurité étant le point le plus important de l’amélioration du projet, nous y avons consacré beaucoup de temps, en particulier sur la mise en place de l’architecture routeur (sur laquelle ont travaillé Gaël Journet et Nathan Ozimek) et de la protection de la connexion contre les attaques en force brute (sur laquelle ont travaillé Jean-François Marcourt et Nathan Ozimek).

Des améliorations quant à l’algorithme de génération des tournois (comme la possibilité de choisir le nombre de phases de poules, ou la prise en compte des résultats précédents de chaque équipe dans la génération des poules) n’ont finalement pas été mises en place par manque de temps et car la sécurisation du projet était notre priorité.

## Architecture routeur

L’architecture routeur a permis de grandement contribuer à la sécurité et qualité logicielle du projet. Lors du lancement du serveur, le site démarre à l’intérieur du dossier webroot, sur la page index.php qui inclut la page routeur app.php à l’aide d’une commande require\_once. L’utilisation du require\_once permet de s’assurer que les visiteurs passent par le routeur, et ne peuvent donc pas contourner l’architecture, tout comme le placement d’index.php à part dans le dossier webroot, d’où il n’est possible d’accéder à aucune page.

La page routeur index.php récupère toute URL entrée par l’utilisateur ou par un lien du site grâce à la variable superglobale $\_SERVER, en utilisant la fonction explode() afin de pouvoir utiliser de potentielles variables passées en GET.

L’URL est ensuite entrée dans un switch, qui redirige vers la page correspondante à l’aide d’un require\_once. Le passage des variables passées en GET nous a d’abord posé problème, car nous ne voyions pas de façon de les faire transiter par le routeur malgré le fait qu’elles étaient cruciales au bon fonctionnement du site. Après avoir découvert et mis en place la fonction explode() pour récupérer l’URL tout en conservant ces variables, ce problème a été résolu.

Les pages de configuration associées à des formulaires, comme celles liées à la création de nouveaux tournois ou utilisateurs, n’ont pas de route spécifique. Lorsque le routeur détecte la route d’une page contenant un formulaire, il vérifie si la méthode est GET (signifiant que l’utilisateur accède simplement à la page) ou POST (signifiant que l’utilisateur a validé le formulaire et que la page s’est rechargée en méthode POST) et renvoie respectivement vers la page du formulaire ou sa page de configuration correspondante.

Les pages de configuration non associées à des formulaires ont leur propre route, car la méthode précédente ne peut pas être utilisée dans leur cas : afin de garantir la sécurité de ces pages sensibles, une vérification du statut de l’utilisateur est faite lors de l’accès à ces pages. Si l’utilisateur n’est pas un administrateur, la page n’effectue aucun traitement et le renvoie à la place immédiatement vers la page d’index.

Dans le cas où un utilisateur entre une URL qui n’existe pas, le routeur renvoie à la page par défaut d’erreur 404.

## Protection de la connexion utilisateur

Dans l’optique d’une mise en production futur il est nécessaire voir primordiale de ne rien lésiner côté sécurité. Et quoi de plus sensible qu’un login sur un site web ?

Pour sécuriser le login nous avions deux possibilités :

* Développer nous même un algorithme permettant de vérifier que dans la même intervalle chronologique un utilisateur ne soumet pas une trop grosse quantité de requête de connexion, si cela arrivait notre algorithme devrais alors bloquer toutes les requêtes provenant de l’adresse IP de connexion de l’utilisateur.
* La deuxième option est de mettre un place une vérification par captcha sur le login, empêchant donc la soumissions de toutes requêtes ne provenant pas d’un réel utilisateur et ainsi prévenant de toute tentative de Brut force.

L’option qui à été mise en application sur notre projet est celle du Captcha, car elle nous permet d’atteindre le même objectif de protection, mais tout en rentrant dans les délais qui nous restait.

Le Captcha mis en place est le Turnstile de Cloudfare, car contrairement à celui de Google, celui de Cloudfare est rapide à passer en temps qu’utilisateur donc ne pénalise par l’ergonomie de l’application et est respectueux des RGPD.

La mise en place à été faites en suivant les documentations officielles et fonctionne de sorte à vérifié l’utilisateur avant que la requête de connexion ne soit envoyée au server, ce qui évite ainsi l’envoie de requêtes inutiles.

Pour résumer le fonctionnement de notre système de connexion, lorsqu’un utilisateur cherche à se connecter, il entre ses identifiants et attend que le captcha ce valide, celui-ci valider il peut alors tester sa connexion en envoyant ses identifiant à notre fonction d’authentification, qui a pour objectif de comparer avec la base de données la véracité des informations de login envoyés. En fonction de la réponse de cette fonction, la connexion est initialisée avec les permissions et rôles associés au compte ou refusé, renvoyant alors l’utilisateurs sur la page de connexion, duquel il est nécessaire de patienter une nouvelle fois la vérification du captcha avant de pouvoir retenter une nouvelle fois de se connecter.

# RÉsultats

[Vous devez présenter ici l’avancement final de votre logiciel (explication texte et capture d’écran ou photos du dispositif en fonctionnement, …). Vous pouvez mettre des comparaisons avec des logiciels concurrents ou de l’évolution par rapport à une évolution. Vous pouvez mettre en avant les caractéristiques techniques comme les performances (temps d’exécution de votre système, la mise ne place d’ergonomie, etc…). Vous devez également mettre en avant l’état actuel de votre produit, cad, vous devez expliciter sous forme de schéma ou de tableau les tâches que vous avez finies, les tâches non résolues ou mises de côté, annoncer les améliorations possibles, etc…]

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

# Conclusion

[Une conclusion qui inclue un bilan sur le travail présenté : l’aspect technique (éventuellement des propositions techniques différentes de celles exposées), professionnel et humain, avec notamment :

* la reprise de vos objectifs définis au début de projet,
* un bilan personnel de chaque membre du groupe en tant que développeur « junior » : les compétences mises en action dans le contexte professionnel, celles acquises au cours de votre BUT (savoirs, savoir-faire, savoir-être et savoir-agir).]

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

# RÉfÉrences

**Aucune source spécifiée dans le document actif.**

Matériel / logiciel / méthode utilisé(e)(s) :

Résumé en français :

Mots clés :