Document d’analyse de Qualité de Développement POST-PROJET

JOURNET Gael
OZIMEK Nathan
MARCOURT Jean-François
S4G3.2 - 2022/2023

Document réalisé en groupe pour le rendu du *14/04/2023.*

I. Introduction / Remise en contexte

Dans le cadre de notre projet universitaire de SAE de 2ème année de BUT Informatique, nous avons été confrontés à de nombreuses difficultés et challenges. Malgré nos efforts, nous avons également commis des erreurs durant le processus de développement. Celles-ci ont eu un impact sur le déroulement du projet, ainsi que sur la qualité du résultat final. Dans le présent document, nous souhaitons revenir sur ces erreurs et aborder les leçons que nous avons pu en tirer, avec pour objectif de pouvoir améliorer la qualité de notre travail pour de futurs projets.

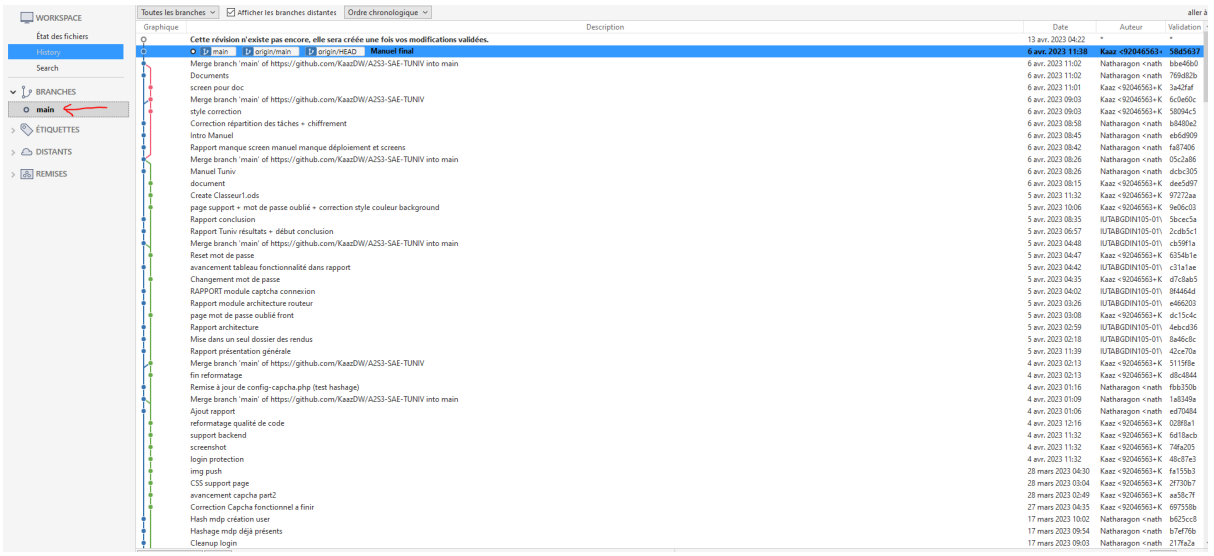
Notre projet s'intitule TUNIV, il s'agit d'une application web réalisée sous la tutelle de Monsieur Buathier avec pour objectif de permettre l'organisation et la gestion de tournois sportifs universitaires, le tout développé en PHP natif (avec SCSS, MySQL et JS).

Le projet a été développé par Gaël Journet, Jean-François Marcourt et Nathan Ozimek, avec Visual Studio Code comme IDE, Notion puis Trello comme outil de coordination et Github comme outil de versionnage.

II. Gestion du Git

II.1 Evaluation du problème

Durant l'entièreté des deux semestres pendant lesquels nous avons travaillé sur la SAÉ, nous n'avons travaillé que sur une seule branche de Git, sans jamais en créer d'autre ou presque. Cela a mené à des problèmes récurrents au cours du développement de notre projet, étant donné que nous travaillions tous sur des fonctionnalités différentes en simultané (ou parfois même sur la même fonctionnalité, voire sur la même page, comme ça a été le cas lors de la mise en place de l'architecture routeur) : comme on pouvait s'y attendre, les conflits de fusion étaient hebdomadaires ou presque, ce qui nous a fait perdre beaucoup de temps qui aurait pu être mieux utilisé.



Source : SourceTree; Ici les différents flux proviennent tous de la branche ***'main'***, juste le contenu sur chacun de nos poste n'était pas le même, puis merge en fin de fonctionnalité.

On peut également citer le cas particulier de la page db.php, qui servait à initialiser la connexion à la base de données dans tout le projet et qui contenait trois versions différentes du code, une par membre de l'équipe. Or, le versionnage constant de la branche unique menait souvent à la situation où la mauvaise version de la branche était active, ce qui nous faisait perdre du temps étant donné que l'on cherchait à résoudre une erreur que nous avions déjà résolu de nombreuses fois.

```php
<?php

// Connection à la base de donnée
// UTILISEZ LES COMMENTAIRES POUR APPELER LA BASE DE DONNÉE CORRESPONDANT AU SUPPORT UTILISÉ.

// // DB connection JF(machine personnelle)
try {
    $pdo = new PDO("mysql:dbname=db_tuniv;host=localhost", "root", "", [PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION]);
} catch (PDOException $e) {
    die();
}

// DB connection Gael (IUT)
// try{
//     $pdo = new PDO ("mysql:dbname=p2106013;host=iutbg-lamp.univ-lyon1.fr", "p2106013", "12106013", [PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION]);
// }catch(Exception $e){
//     die();
// }

// DB connection Nathan (IUT)
// try {
//     $pdo = new PDO("mysql:dbname=p2106229;host=iutbg-lamp.univ-lyon1.fr", "p2106229", "12106229", [PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION]);
// } catch (Exception $e) {
//     die();
// }
```

II.2 Solution potentielle ou appliquée

Les cours de qualité de développement lors desquels nous avons eu l'occasion d'apprendre à réellement optimiser l'utilisation de Github sur un projet de groupe comme celui-ci n'ayant eu lieu qu'en fin d'année, il n'a pas été possible pour nous d'appliquer ces connaissances plus tôt. Cependant, si c'était à refaire nous nous y prendrions totalement différemment, en créant une branche par fonctionnalité, correctement séparées et nommées, afin que chacun puisse travailler de son côté et que les fusions soient propres et sans problème, sans oublier des commits et pushs réguliers et nommés correctement afin de pouvoir constamment garder une trace de notre avancée et ne pas perdre de travail..

III. Lisibilité du code

III.1 Evaluation du problème

L'utilisation d'un code pas propre dans notre SAE TUNIV a entraîné de nombreuses conséquences négatives pour le projet dans son ensemble.

Des copier-coller trop larges de parties de codes ont introduit des erreurs et des incohérences dans le projet, avec par exemple des fonctions ou des fichiers de configurations appelées sur des pages qui n'en avaient pas l'utilité et dont personne ne connaissait les raisons de la présence. C'est là, ça ne sert à rien, et aucun d'entre nous ne sait expliquer pourquoi.

En parallèle de ça, du code non documenté a compliqué la compréhension de l'ensemble du projet lors du développement de fonctions similaires ou associées ou encore lors de phases de débogage, en particulier pour les membres du groupe qui n'avaient pas été impliqués dans l'écriture de ses fonctions.

Un manque de concertation au niveau du Code Style a également mené à de nombreuses incohérences, avec la présence de franglais ou des combinaisons de noms en camelcase et utilisant des underscores.

```
14  }
15  $sport = $pdo->quote($_POST["sport"]);
16
17
18  if (empty($_POST["name-capitaine"])){
19      $_SESSION["equipeErreur"] = "Nom du capitaine manquant";
20      header("Location: /dashb-admin.php");
21  }
22  $nom_cap = $pdo->quote($_POST["name-capitaine"]);
23
24  if (empty($_POST["firstname-capitaine"])){
25      $_SESSION["equipeErreur"] = "Prenom du capitaine manquant";
26      header("Location: /dashb-admin.php");
27  }
28  $prenom = $pdo->quote($_POST["firstname-capitaine"]);
29
30  $users = $pdo->prepare('SELECT ID_User from Utilisateurs where Nom =:varnom and Prenom=:varprenom');
31
32  $users->execute(
33      [
34          'varnom' => $nom_cap,
35          'varprenom' => $prenom,
36      ]
37  );
38  $user = $users->fetch();
39
40  if($user){
41      $id_user = $user[0];
42  } else{
43      $id_user = 1;
44  }
45
46  $id_user=$pdo->quote($id_user);
47
48  $sql = "INSERT INTO Equipe VALUES (0, $nom, $sport, $id_user);";
49  $res = $pdo->exec($sql);
50  if (!$res) {
51      $_SESSION["equipeErreur"] = "La création de l'équipe a échoué, veuillez réessayer. Si l\'erreur persiste, contactez le support.";
52      header("Location: /dashb-admin");
53  } else {
54      $sql = "SELECT max(ID_Tournoi) FROM Tournoi;";
55      $id = $pdo->query(($sql))->fetch()[0];
56  }
57
58  header("Location: /dashb-admin");
59
```

exemple de code, avant correction de notre part nous avions des fichiers entiers de ce style, sans commentaires, des lignes répartis sur 7 etc..

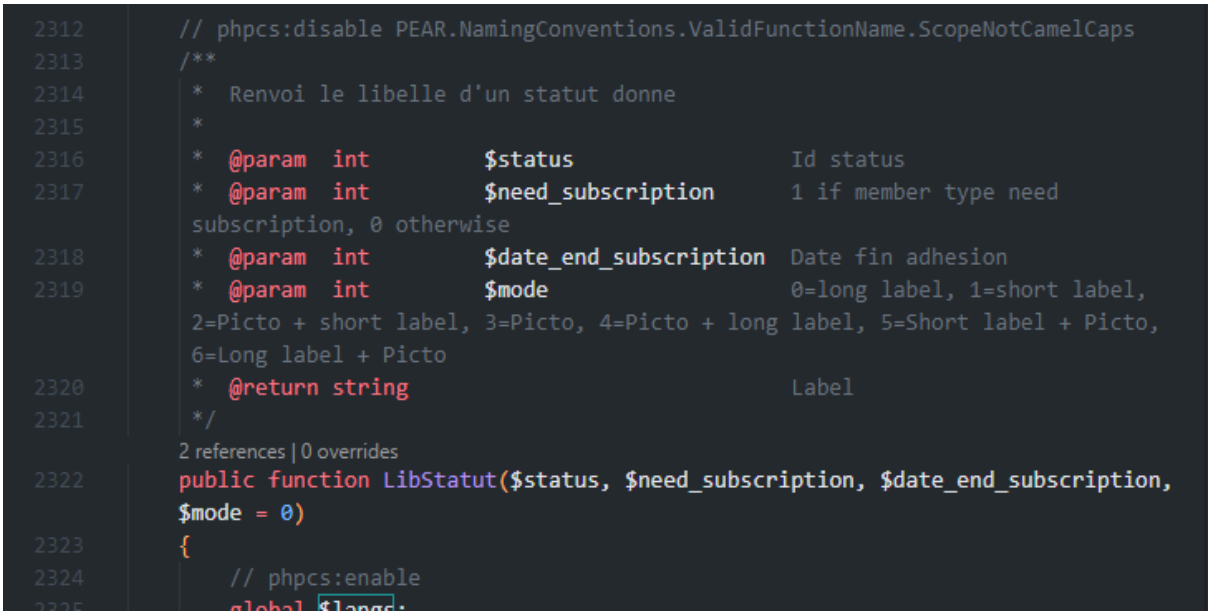
Tout cela cumulé a donc grandement augmenté le temps nécessaire pour effectuer des ajouts ou de la maintenance sur le projet. En somme, le développement d'un code pas suffisamment propre nous a prouvé à quel point cela peut rendre la collaboration difficile et affecter la qualité et la performance du projet dans son ensemble.

III.2 Solution potentielle ou appliquée

Pour ce type de problème, la solution n'est pas sorcier, il faut tout revoir, retourner sur une base saine, pour pouvoir continuer de développer dans des conditions optimales. Pour ce faire nous avons réparti les tâches avec, dans un premier temps, une review complète de l'ensemble du projet avec pour objectif : reformatage (pour que tout soit clair et lisible d'un simple coup d'oeil), suppression du code inutile ou encore renommage ou suppression de certaines variables temporaires abandonnées. Ceci fait chacun d'entre nous a alors pu refaire le tour des fichiers un à un avec pour finalité de commenter les parties de code non explicites qu'il a pu développer.

```php
include("db.php");

$tournoi = $pdo->prepare('SELECT * FROM Tournoi WHERE ID_Tournoi =:varId');
$tournoi->execute(['varId' =>$_GET["id"]]);
$tournoi=$tournoi->fetchAll();

if ($tournoi[0]["Etape"]!=1){
    header("Location: ../pages/match-tournois.php?id=".$_GET["id"]);
}

// On bloque les matchs de poules
$changerEtat = $pdo->prepare("UPDATE MatchTournoi SET Etat=1 WHERE ID_Tournoi = :varId;");
$changerEtat->execute(['varId' => $_GET["id"]]);

// Calcul du nombre d'équipes restantes
$nbEquipes = $pdo->prepare("SELECT count(*) FROM Participer WHERE ID_Tournoi = :varId");
$nbEquipes->execute(['varId' => $_GET['id']]);
$nbEquipes = $nbEquipes->fetch()[0];

$nbFinalistes = 2;
while ($nbFinalistes*2<$nbEquipes){
    $nbFinalistes = $nbFinalistes*2;
}

// Création du tableau des équipes finalistes
$listeFinalistes = $pdo->prepare("SELECT ID_Equipe FROM Participer WHERE ID_Tournoi = :varId ORDER BY Score DESC");
$listeFinalistes->execute(['varId' => $_GET['id']]);
$listeFinalistes = $listeFinalistes->fetchAll();

$tabFinalistes = [];

for ($i=0; $i<$nbFinalistes; $i++){
    $tabFinalistes[$i] = $listeFinalistes[$i];
}

//Création des matchs
$sport = $pdo->prepare("SELECT Sport FROM Tournoi WHERE ID_Tournoi = :varTournoi;");
$sport->execute(['varTournoi' => $_GET["id"]]);
$sport = $sport->fetch()[0];
$sport = $pdo->quote($sport);
$creerMatch = $pdo->prepare("INSERT INTO MatchTournoi VALUES(0, $sport, now(), now(), 'A_définir', 1, :varTournoi, 0);");

```

exemple de code commenté et reformaté correctement : compréhensible et maintenable.



```
2312    // phpcs:disable PEAR.NamingConventions.ValidFunctionName.ScopeNotCamelCaps
2313    /**
2314     *  Renvoi le libelle d'un statut donne
2315     *
2316     *  @param  int         $status                 Id status
2317     *  @param  int         $need_subscription      1 if member type need
        subscription, 0 otherwise
2318     *  @param  int         $date_end_subscription  Date fin adhesion
2319     *  @param  int         $mode                   0=long label, 1=short label,
        2=Picto + short label, 3=Picto, 4=Picto + long label, 5=Short label + Picto,
        6=Long label + Picto
2320     *  @return string                              Label
2321     */
        2 references | 0 overrides
2322    public function LibStatut($status, $need_subscription, $date_end_subscription,
        $mode = 0)
2323    {
2324        // phpcs:enable
            global $langs;
```

Également, mettre en place ce type de commentaire (type: doxygen) permettrait à l'IDE d'afficher de lui-même les informations relatives à nos fonctions d'un simple survole.

Si c'était à refaire, ou lors de nos prochains projets, la mise en place d'automatismes de commentaire et une communication accrue serait à appliquer. Le tout en parallèle de l'outil de versionnage et de travail collaboratif Github cela devrait pouvoir pallier ces problèmes de qualité de code.

IV. Tests unitaires

IV.1 Evaluation du problème

Notre projet ne contient pas de Test Unitaires, or cela aurait pu être bénéfique sur différents points. Tout d'abord cela à entraîner une phase de test plus longue car à chaque ajout ou modification du code nous devions vérifier manuellement si tout fonctionnait correctement et si nous n'avions pas créer de nouveau problèmes. De plus, nous avons réalisé des fonctionnalités au premier abord fonctionnelles mais qui plus tard se sont révélées défaillantes selon certaines actions de l'utilisateur.

IV.2 Solution potentielle ou appliquée

C'est durant le cours de qualité de développement que nous nous sommes aperçus que les tests unitaires auraient pu nous permettre de gagner du temps et de rendre un projet avec une logique plus sûre et fonctionnelle dans tous les cas.

En effet, les tests unitaires sont une pratique de programmation qui consiste à écrire des tests automatisés pour chaque composant logiciel, ou "unité", afin de vérifier leur bon fonctionnement indépendamment du reste du code. Cette approche permet de détecter rapidement les erreurs et les bugs, d'améliorer la qualité du code et de faciliter sa maintenance.

En intégrant les tests unitaires dès le début de notre projet, nous aurions pu éviter les problèmes causés par des bugs survenus entre deux ajouts et améliorer notre intégration continue (test en local car non hébergé) tout en bénéficiant d'une meilleure organisation générale de notre avancement. De plus, les tests unitaires nous auraient permis de fixer certaines priorités de code et exigences du projet ainsi que d'identifier les éventuels problèmes dès le début du processus de développement.

En somme, les tests unitaires sont une pratique essentielle pour assurer la qualité et la robustesse d'un projet logiciel, et leur intégration dès le début aurait pu faire la différence sur les points que je viens d'énoncer ainsi que nous donner un élément supplémentaire prouvant l'efficacité du travail fourni.

V. Documentation

V.1 Evaluation du problème

Il est fréquent en informatique de devoir aborder des sujets que nous ne connaissons pas ou trop peu. Cela peut notamment concerner des aspects de sécurité, où il est essentiel de protéger l'application contre les vulnérabilités et les attaques.

Dans ces cas-là, il nous semblait pertinent de d'abord nous concentrer sur le développement des fonctionnalités dans leur ensemble, avant de se pencher sur les problèmes de sécurité et les façons de s'en protéger. Notre objectif était avant tout de chercher à répondre aux besoins du client : on fait d'abord, on améliore après.

Le problème a été le manque total de documentation et de connaissance lors de cette phase de développement. Le fait est que cela a nécessité une réécriture complète de certaines parties du code et un travail supplémentaire qui aurait pu être évité en abordant les problèmes de sécurité plus tôt.

A titre d'exemple, cela n'a pas été un problème concernant la protection contre les failles XSS, car il n'est pas très contraignant de s'en protéger, notre projet restant un projet à notre échelle, très rapidement les protection minimales étaient en place.

Mais pour le login, c'est tout le contraire, nous aurions pu être davantage productifs en nous étant documentés au préalable.

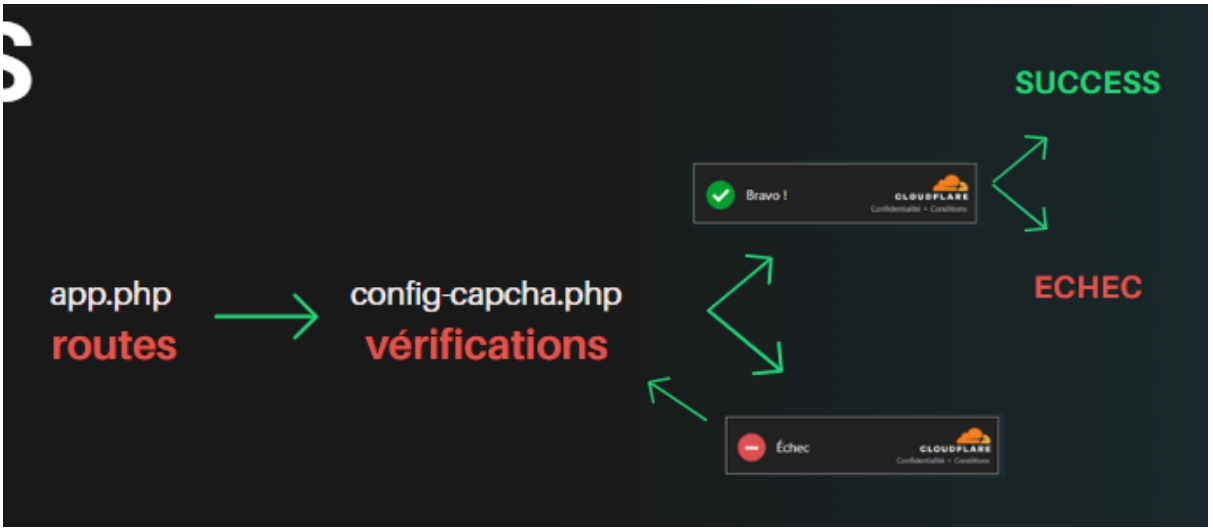


Schéma présenté lors de notre soutenance démontrant globalement comment notre login fonctionne après la mise en place du captcha. La vérification SUCCESS/ECHEC de fin correspondant approximativement à ce que nous avions développé à la base.

V.2 Solution potentielle ou appliquée

Nous savions que nous devions mettre cette sécurité en place dès le début du projet, alors se renseigner sur les méthodes à utiliser dès son développement nous aurait permis de gagner du temps et de réduire les risques d'erreurs. En effet, lors de la phase de développement, nous avons dû procéder à plusieurs réécritures et réévaluation de la logique derrière notre système d'authentification pour sécuriser celui-ci à l'aide d'un captcha. Cette démarche aurait pu être évitée si nous avions pris le temps de nous documenter et de nous former sur les méthodes de sécurisation des systèmes d'authentification.

Il est donc essentiel de comprendre l'importance de la sécurité dès le début du projet, de se documenter et de se former en conséquence, afin d'adopter des pratiques sécurisées dès le début de la phase de développement. Cela nous aurait permis d'optimiser davantage notre temps de travail.

VI. Conclusion

En conclusion, malgré les difficultés et les erreurs rencontrées durant le processus de développement de notre projet universitaire de 2ème année de BUT Informatique, nous avons pu tirer des leçons importantes du cours de qualité de développement afin d'améliorer notre qualité de travail pour les projets futurs.

- Nous avons réalisé que l'intégration des tests unitaires dès le début du projet aurait pu nous permettre d'éviter de nombreux problèmes et de gagner du temps.
- Nous avons constaté la nécessité d'un code propre et maintenable
- Nous avons observé l'utilité et l'étendue des possibilités offertes par un outil de versionnage et de travail collaboratif comme Github.
- Nous avons également compris l'importance de bien se documenter avant de se lancer dans la programmation.

Malgré ces erreurs, nous sommes fiers du résultat final de notre application web TUNIV, qui permet l'organisation et la gestion de tournois sportifs universitaires. Nous avons pu mettre en pratique les connaissances acquises tout au long de notre formation en utilisant des technologies telles que PHP, SCSS, MySQL et JavaScript.

Nous espérons que les enseignements tirés de cette expérience nous permettront de développer des projets encore plus performants et de qualité bien supérieure à l'avenir.