Raphaël Lopes

CFPT-INFORMATIQUE

Avril 2019

Documentation TECHNIQUE

TALES OF THE TAVERN



TRAVAIL PRATIQUE INDIVIDUEL 2019

Experts : M. Pascal COURT, M. Borys FOLOMIETOW

maitre d’apprentissages : Mme Katia MOTA STROPPOLO

Table des matières

[1 Table des versions 3](#_Toc6408638)

[2 Introduction 3](#_Toc6408639)

[2.1 Généralités 3](#_Toc6408640)

[3 Cahier des charges 3](#_Toc6408641)

[3.1 Objectif 3](#_Toc6408642)

[3.2 Fonctionnalités 4](#_Toc6408643)

[4 Planification 5](#_Toc6408644)

[4.1 Planification prévue 5](#_Toc6408645)

[4.2 Planification effective 5](#_Toc6408646)

[5 Méthodologie 6](#_Toc6408647)

[5.1 Planification en 6 étapes 6](#_Toc6408648)

[5.2 Développement agile / BACKLOG 6](#_Toc6408649)

[6 Analyse organique 8](#_Toc6408650)

[6.1 Base de données 8](#_Toc6408651)

[6.1.1 Informations 8](#_Toc6408652)

[6.2 MCD 8](#_Toc6408653)

[6.3 MLD 9](#_Toc6408654)

[6.4 Dictionnaire de données 9](#_Toc6408655)

[6.5 Architecture 12](#_Toc6408656)

[TalesOfTheTavern/ 13](#_Toc6408657)

[TalesOfTheTavern/Vue 13](#_Toc6408658)

[TalesOfTheTavern/Vue/bootstrap-4.3.1-dist 17](#_Toc6408659)

[TalesOfTheTavern/Vue/Img 17](#_Toc6408660)

[TalesOfTheTavern/Controleur 18](#_Toc6408661)

[TalesOfTheTavern/Model 19](#_Toc6408662)

[6.6 Analyse de plusieurs fonctionnalités principales 19](#_Toc6408663)

[6.6.1 Découpage MVC 19](#_Toc6408664)

[6.7 Moyenne utilisateur 20](#_Toc6408665)

[7 Framework 21](#_Toc6408666)

[7.1 Bootstrap 21](#_Toc6408667)

[7.2 Font Awesome 21](#_Toc6408668)

[8 Test et plan de test 23](#_Toc6408669)

[9 Conclusion 26](#_Toc6408670)

[9.1 Difficultés rencontrées 26](#_Toc6408671)

[9.2 Amélioration possible 26](#_Toc6408672)

[9.3 Remerciements 26](#_Toc6408673)

[10 Bibliographie 27](#_Toc6408674)

[11 Annexes 27](#_Toc6408675)

[11.1 Glossaire 27](#_Toc6408676)

[11.2 Énoncer 27](#_Toc6408677)

[11.3 Journal de bord 27](#_Toc6408678)

[11.4 Résumé TPI 27](#_Toc6408679)

[11.5 Code source 27](#_Toc6408680)

# Table des versions

Tableau 1 : table des versions

|  |  |  |
| --- | --- | --- |
| Date | Modification | Version |
| 02.04.19 | Création du document | 0.1 |
| 02.04.19 | Page de garde, Introduction, Planification, Cahier des charges | 0.2 |
| 03.04.19 | Analyse organique, Base de données, Dictionnaire de données | 0.3 |
| 04.04.19 | Architecture | 0.4 |
| 10.04.19 | Test et plan de test, Analyse de plusieurs fonctionnalités | 0.5 |
| 15.04.19 | Conclusion, librairies, Annexes | 0.6 |
| 17.04.19 | Planning, Code, Correction orthographe | 1.0 |

# Introduction

## Généralités

Dans le cadre du travail pratique individuel (TPI), j’ai dû réaliser un projet dans le domaine de l'informatique qui consiste d’une application web.

La fonction principale de l’application contient des utilisateurs (connecté à un compte) qui peuvent mettre en ligne des histoires avec une image et que par la suite d’autres utilisateurs peuvent les regarder et le noter sur quatre critères plus la moyenne des 5. Un utilisateur peut rechercher des histoires par son titre et son auteur. Enfin, un utilisateur connecter peut ajouter une histoire à ses favoris.

Note :

1. Styles.
2. Histoire
3. Orthographe
4. Originalité
5. Moyenne des 4 autres

# Cahier des charges

## Objectif

Développer un site web permettant de s’y connecter avec un compte, de pouvoir créer des histoires, noter des histoires que les autres utilisateurs ont créées, ajouter les histoires aux favoris et pouvoir rechercher par titre ou par auteur.

## Fonctionnalités

Les fonctionnalités principales sont :

* Créer un compte / se connecter
* Créer une histoire / afficher l’histoire
* Pouvoir noter une histoire
* Rechercher une histoire
* Ajouter aux favoris une histoire (si on est authentifié)
* Pouvoir voire les histoires les plus récentes (date) ou les mieux noter (moyenne)

Une histoire contient :

* Un titre
* Un auteur
* Une date de création
* Une histoire
* Une note dans 4 différents critères :
  + Style
  + Histoire
  + Orthographe
  + Originalité
* Une note générale qui est la moyenne des 4.
* La catégorie de l’histoire
* Une image (s’il n’y a pas d’image alors on affiche celle de la catégorie)

L’application prévoit 3 types d’utilisateurs

* L’anonyme : c’est un utilisateur qui n’est pas connecté. Il peut seulement lire des histoires et les noter.
* Le membre : c’est un utilisateur qui s’est connecté au préalable. Il a les mêmes droits que « l’anonyme », mais peut en plus, afficher et gérer depuis la page d’accueil une liste d’histoire favorite et, il a une page personnelle permettant de gérer ses informations
  + Nom d’utilisateur (lire, modifier).
  + Email
  + Mot de passe
  + Note générale
  + Toutes les histoires créées par l’utilisateur
* L’administrateur : en plus des droits ci-dessus l’administrateur doit gérer la base de données avec l’interface d’administration de la base de données et, de maintenir le code source de l’application.

# Planification

## Planification prévue

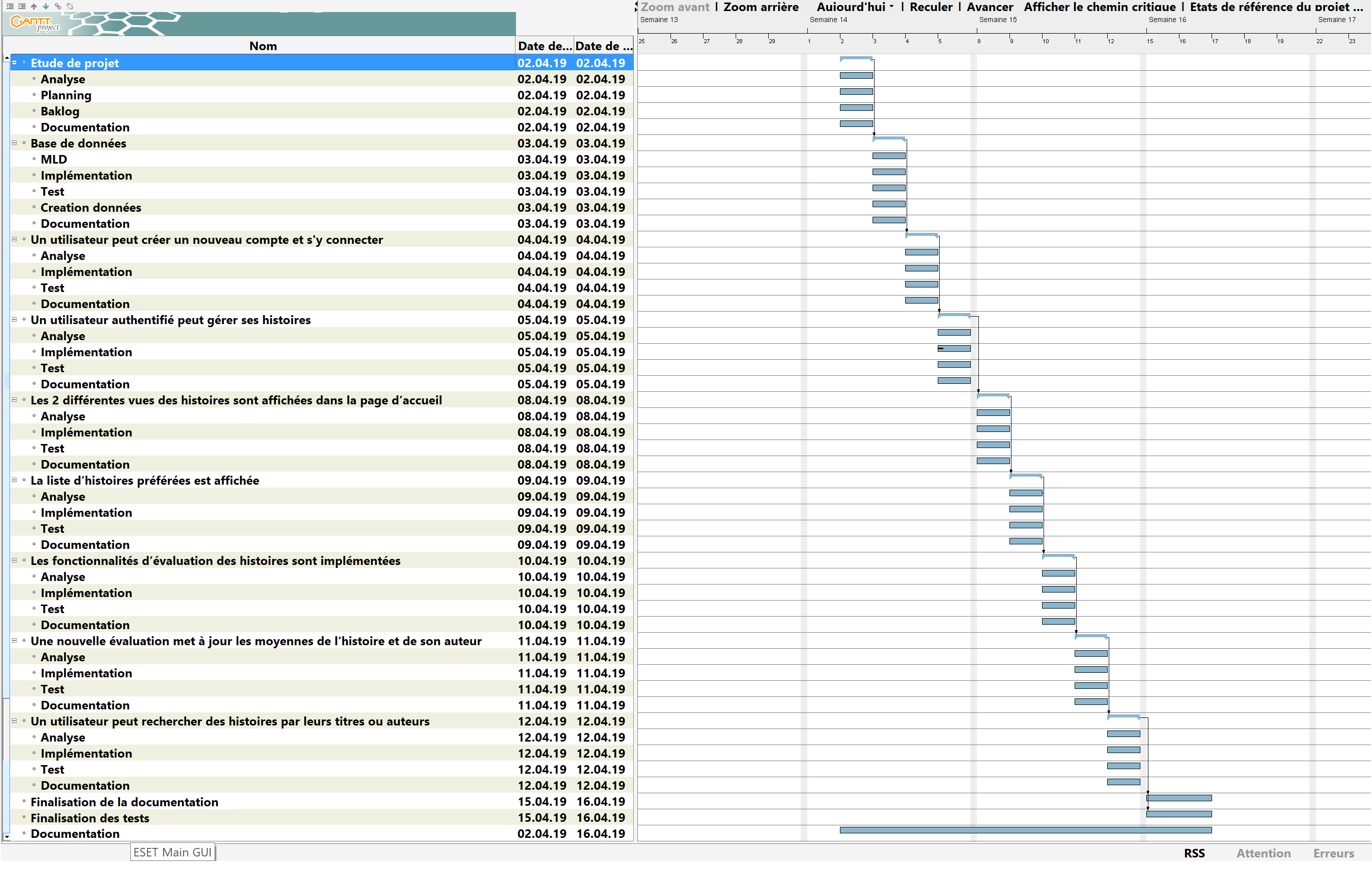


Figure 1 : Planning prévu

## Planification effective

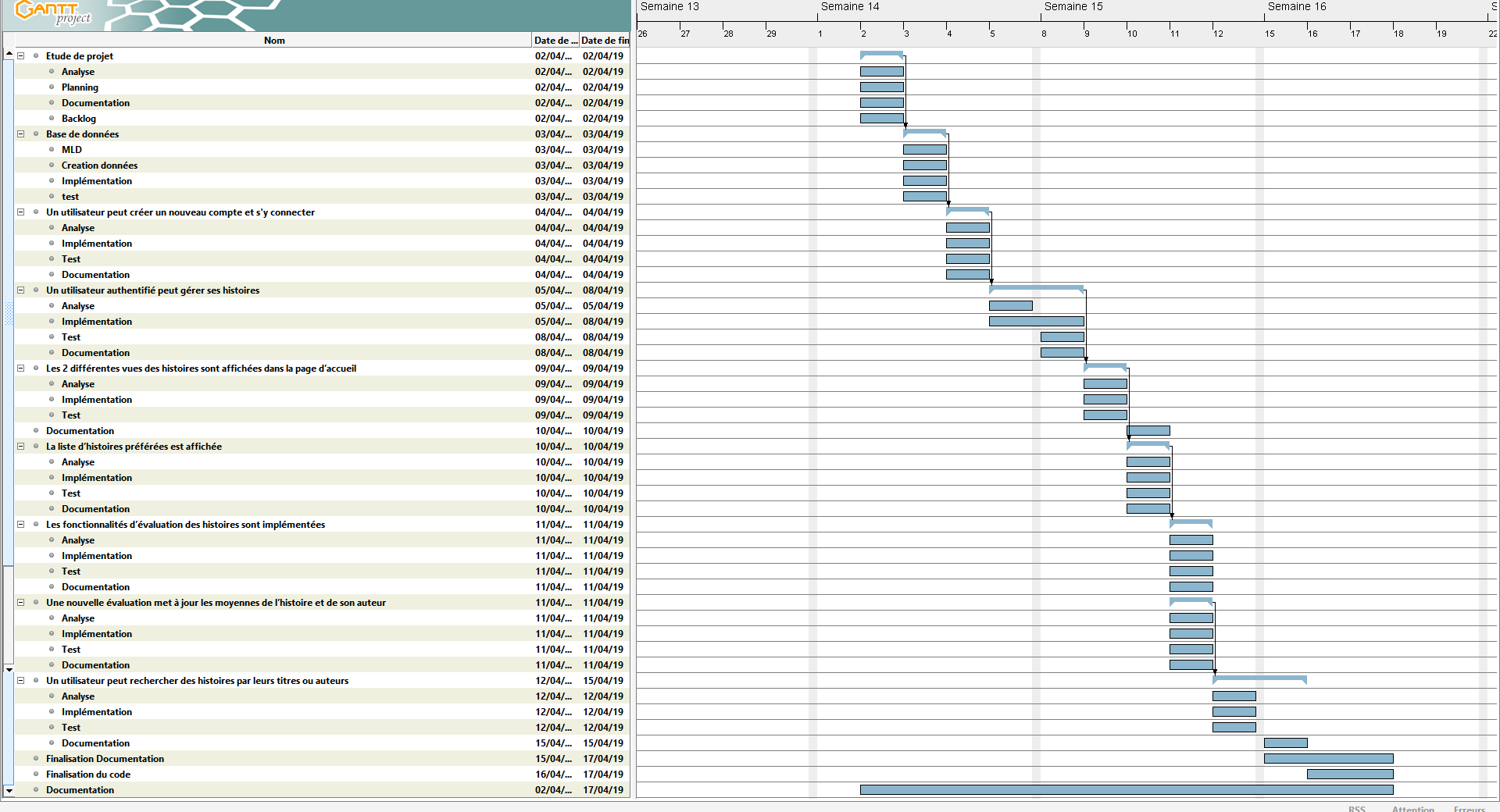


Figure 2 : Planning effectif

J’ai pu suivre mon planning prévu or, pour la tâche « Un utilisateur authentifié peut gérer ses histoires » on peut voir que j’ai eu des problèmes d’implémentation et donc j’ai pris du retard sur mon planning. J’ai pu petit à petit rattraper ce retard.

# Méthodologie

## Planification en 6 étapes

La méthodologie en 6 étapes aide à exécuter, organiser et faire avancer le travail

Elle se divise comme son nom l’indique en 6 étapes :

* Informer : consiste à comprendre et enregistrer la tâche pour se faire une idée et bien cerner le sujet et avoir le maximum d’information
* Planifier : cette étape consiste à répartir les tâches, choisir les tâches et estimer le temps nécessaire.
* Décider : lorsque l’information et de la planification sont faites, il faut prendre une décision.
* Réaliser : cette étape est la partie principale d’une tâche, mais il faut néanmoins respecter le plan que nous avons créé à la planification et ne pas le modifier (si possible)
* Contrôler : chaque travail fait doit être contrôlé, il faut donc le relire, comparer avec le cahier des charges, mesurer et estimer.
* Évaluer : enfin, il faut évaluer tout ce que ne nous avons fait pour savoir si le travail effectuer est bon ou non. Savoir où nous avons perdu du temps, fait des erreurs, si tout s’est bien passé, etc.

J’ai utilisé cette méthodologie dans la panification dans le GANTT pour chaque tâche.

## Développement agile / BACKLOG

Pour chaque critère, j’ai décidé de la prendre et la transformer en user stories. Ensuite chaque jour je fais une « sorte » de sprint avec l’user stories du jour et je l’implémente. J’ai utilisé cette méthodologie tout le long du développement.

Tableau 2 : User stories

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | En tant que … | Je souhaite… | Pour | Résultat obtenue | Statut |
| A14 | Anonyme | Créer un nouveau compte et pourvoir m'y connecter | Avoir une page "création du compte" fonctionnelle et une page de "login" qui permet de se connecter |  | À faire |
| A15 | Admin/Membre | Avoir accès à ma page de gestion d'histoire | Avoir une page "mon compte" qui affiche les informations du compte (du nom d'utilisateur aux histoires qu'il a créé). |  | À faire |
| A16 | Anonyme/Admin/Membre | Avoir 2 tris sur la page accueille 1-moyenne de notes (tri décroissant) 2-les histoires plus récentes (tri décroissant) | Pouvoir afficher les histoires sur la page d'accueil avec 2 tris différents. 1-moyenne de notes (tri décroissant) 2-les histoires plus récentes (tri décroissant) |  | À faire |
| A17 | Admin/Membre | Avoir mes histoires que j'ai marquées en favoris sur la page accueil.php et pouvoir les ajouter/ supprimer | Afficher sur la page principale des utilisateurs identifier leur histoire favorite et pouvoir les ajouter / supprimer |  | À faire |
| A18 | Anonyme/Admin/Membre | Pouvoir évaluer une histoire selon le style, l'histoire, orthographe et l'originalité. | Permettre à n'importe quel utilisateur de pouvoir noter une histoire |  | À faire |
| A19 | Anonyme/Admin/Membre | Que lorsque j'évalue une histoire sa note et la note de l'auteur changent | Que les moyennes des histoires / auteur changent. |  | À faire |
| A20 | Anonyme/Admin/Membre | Pouvoir chercher une histoire par son titre ou par son auteur | Permettre aux utilisateurs de chercher une histoire avec le titre ou l’auteur |  | À faire |

# Analyse organique

## Base de données

### Informations

Compte Admin :

Nom d’utilisateur : AdminTalesTPI  
Mot de passe : SuperTales19

PhpMyAdmin : version 17.0

PHP : version 5.6.30

Apache : version 2.4.25

MySQL : version 5.7.17

Moteur de base de données : InnoDB

Interclassement : utf8mb4\_general\_ci

## MCD

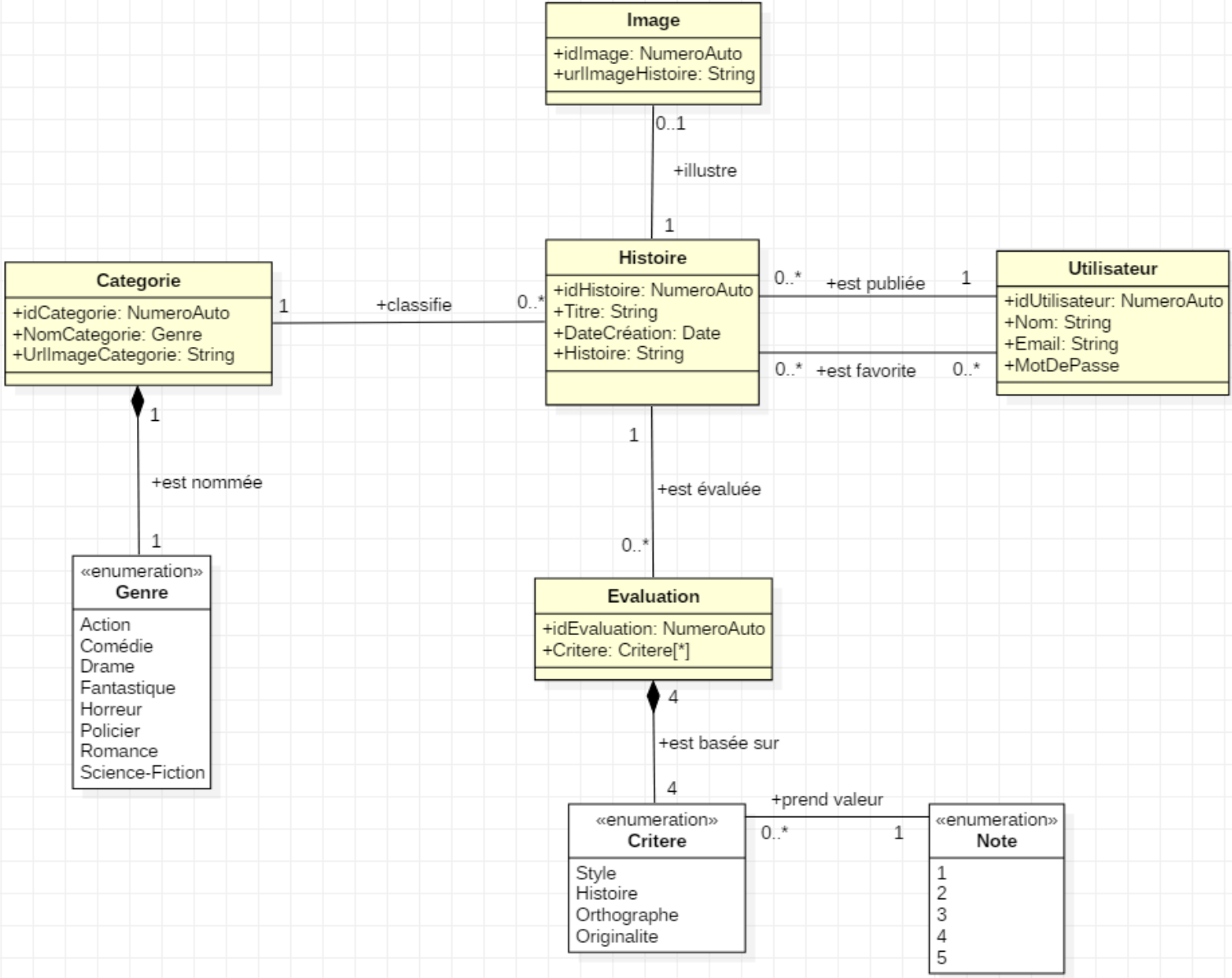


Figure 3 : Model conceptuelle de donnée

## MLD

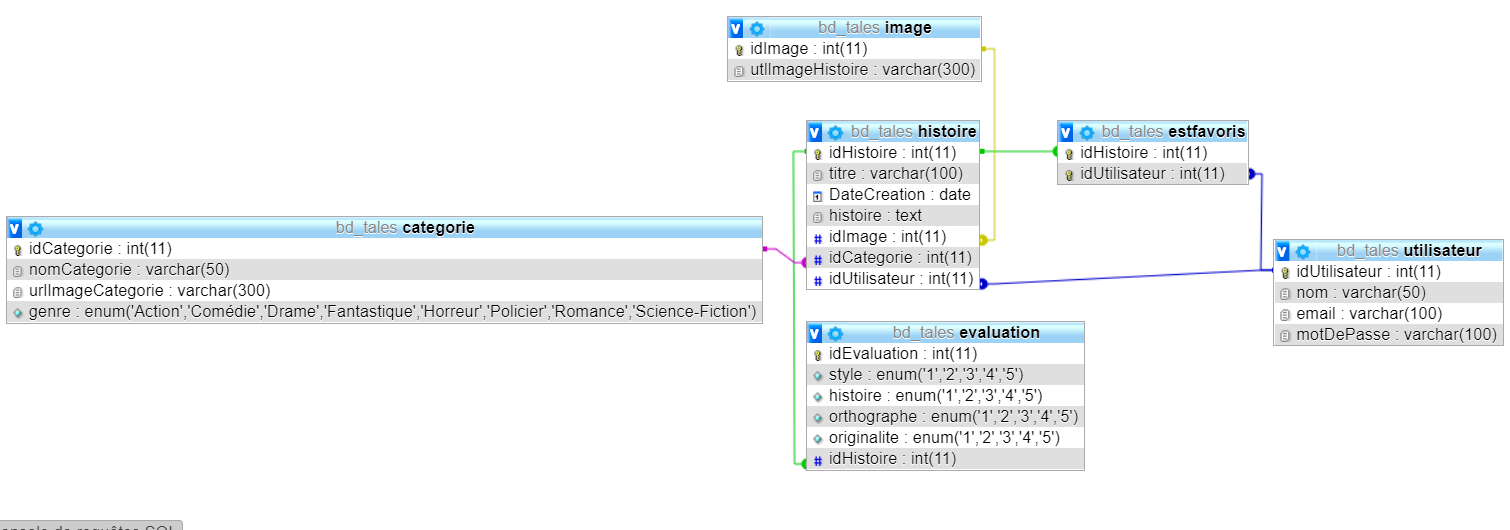


Figure 4 : Model logique de données

## Dictionnaire de données

**categorie**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Colonne | Type | Null | Valeur par défaut | Est relié à | Commentaires | MIME |
| idCategorie (Primaire) | int(11) | Non |  |  |  |  |
| nomCategorie | varchar(50) | Non |  |  |  |  |
| urlImageCategorie | varchar(100) | Non |  |  |  |  |

Index

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Nom de l'index | Type | Unique | Compressé | Colonne | Cardinalité | Interclassement | Null | Commentaire |
| PRIMARY | BTREE | Oui | Non | idCategorie | 8 | A | Non |  |

**estfavoris**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Colonne | Type | Null | Valeur par défaut | Est relié à | Commentaires | MIME |
| idHistoire (Primaire) | int(11) | Non |  | histoire -> idHistoire |  |  |
| idUtilisateur (Primaire) | int(11) | Non |  | utilisateur -> idUtilisateur |  |  |

Index

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Nom de l'index | Type | Unique | Compressé | Colonne | Cardinalité | Interclassement | Null | Commentaire |
| PRIMARY | BTREE | Oui | Non | idHistoire | 3 | A | Non |  |
| idUtilisateur | 4 | A | Non |
| idHistoire | BTREE | Non | Non | idHistoire | 3 | A | Non |  |
| idUtilisateur | BTREE | Non | Non | idUtilisateur | 2 | A | Non |  |

**evaluation**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Colonne | Type | Null | Valeur par défaut | Est relié à | Commentaires | MIME |
| idEvaluation (Primaire) | int(11) | Non |  |  |  |  |
| style | enum('1', '2', '3', '4', '5') | Non |  |  |  |  |
| histoire | enum('1', '2', '3', '4', '5') | Non |  |  |  |  |
| orthographe | enum('1', '2', '3', '4', '5') | Non |  |  |  |  |
| originalite | enum('1', '2', '3', '4', '5') | Non |  |  |  |  |
| idHistoire | int(11) | Non |  | histoire -> idHistoire |  |  |

Index

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Nom de l'index | Type | Unique | Compressé | Colonne | Cardinalité | Interclassement | Null | Commentaire |
| PRIMARY | BTREE | Oui | Non | idEvaluation | 2 | A | Non |  |
| idHistoire | BTREE | Non | Non | idHistoire | 2 | A | Non |  |

**histoire**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Colonne | Type | Null | Valeur par défaut | Est relié à | Commentaires | MIME |
| idHistoire (Primaire) | int(11) | Non |  |  |  |  |
| titre | varchar(100) | Non |  |  |  |  |
| dateCreation | date | Non |  |  |  |  |
| histoire | text | Non |  |  |  |  |
| idImage | int(11) | Oui | *NULL* | image -> idImage |  |  |
| idCategorie | int(11) | Non |  | categorie -> idCategorie |  |  |
| idUtilisateur | int(11) | Non |  | utilisateur -> idUtilisateur |  |  |

Index

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Nom de l'index | Type | Unique | Compressé | Colonne | Cardinalité | Interclassement | Null | Commentaire |
| PRIMARY | BTREE | Oui | Non | idHistoire | 2 | A | Non |  |
| idImage | BTREE | Non | Non | idImage | 2 | A | Oui |  |
| idCategorie | BTREE | Non | Non | idCategorie | 2 | A | Non |  |
| idUtilisateur | BTREE | Non | Non | idUtilisateur | 2 | A | Non |  |

**image**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Colonne | Type | Null | Valeur par défaut | Est relié à | Commentaires | MIME |
| idImage (Primaire) | int(11) | Non |  |  |  |  |
| urlImageHistoire | varchar(100) | Non |  |  |  |  |

Index

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Nom de l'index | Type | Unique | Compressé | Colonne | Cardinalité | Interclassement | Null | Commentaire |
| PRIMARY | BTREE | Oui | Non | idImage | 4 | A | Non |  |

**utilisateur**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Colonne | Type | Null | Valeur par défaut | Est relié à | Commentaires | MIME |
| idUtilisateur (Primaire) | int(11) | Non |  |  |  |  |
| Nom | varchar(50) | Non |  |  |  |  |
| Email | varchar(100) | Non |  |  |  |  |
| motDePasse | varchar(100) | Non |  |  |  |  |

Index

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Nom de l'index | Type | Unique | Compressé | Colonne | Cardinalité | Interclassement | Null | Commentaire |
| PRIMARY | BTREE | Oui | Non | idUtilisateur | 3 | A | Non |  |
| email | BTREE | Oui | Non | email | 3 | A | Non |  |

## Architecture

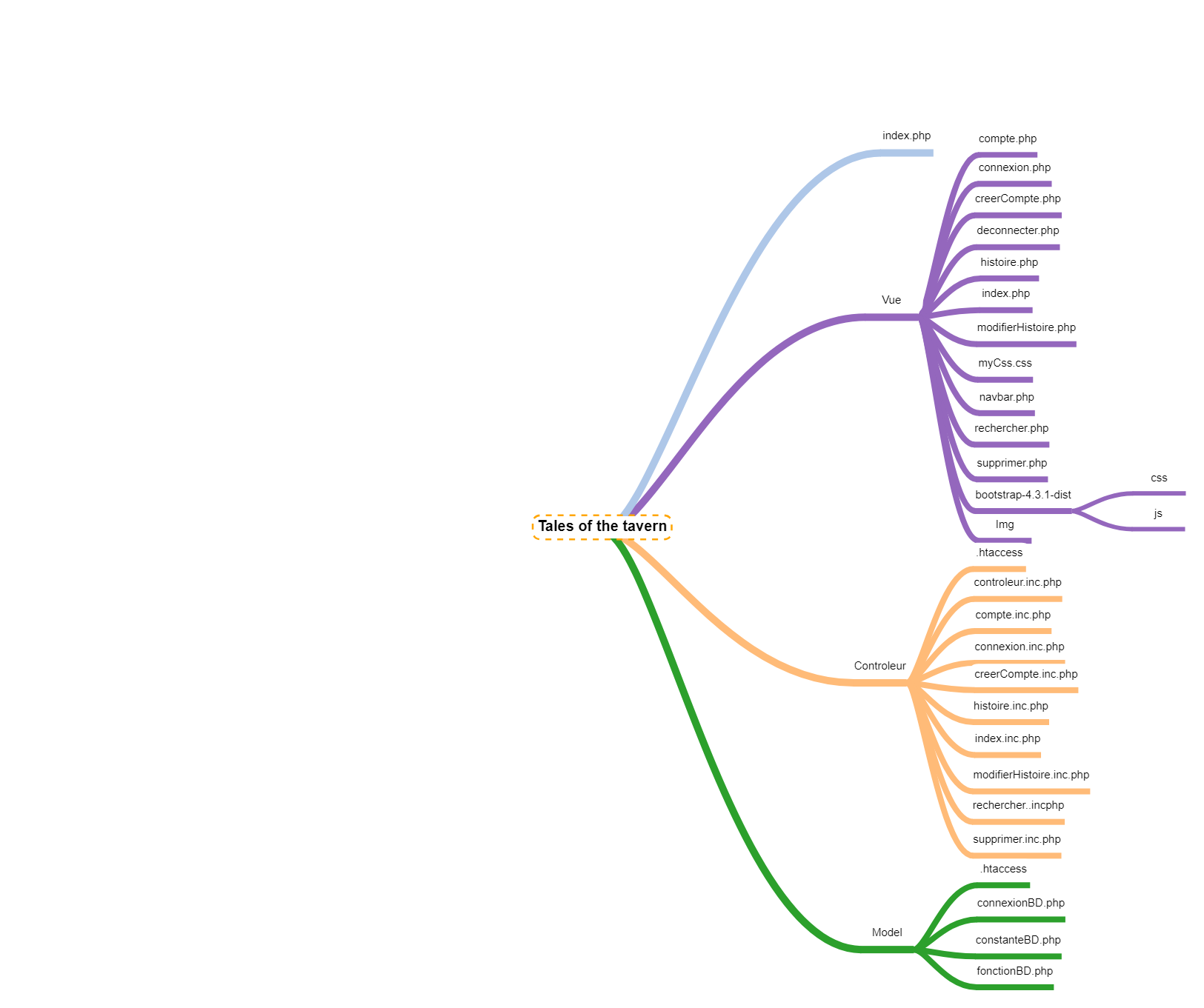


Figure 5 : Architecture du site web

Le site web se divise en trois parties la vue, le contrôleur et le modèle (découpage MVC) lorsque l’utilisateur fait une action sur la vue, elle appelle le contrôleur qui va traiter l’information, si le contrôleur a besoin des informations stocker dans la base alors, le contrôleur fait appelle au modèle qui va lui retourner les informations de la base qui transforme ses informations en information utilisable pour la vue.

### TalesOfTheTavern/

Ceci est le dossier parent du site.

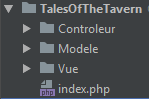


Figure 6 : Architecture dossier parent

* index.php : cette page redirige l’utilisateur vers Vue/index.php

### TalesOfTheTavern/Vue

Ce dossier ne fait qu’afficher les informations et en cas d’action il envoie les données à traiter au diffèrent contrôleur.

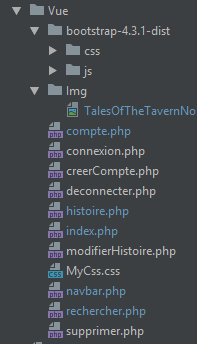


Figure 7 : Architecture Vue

* compte.php : Cette page permet d’afficher les champs pour la modification des informations de l’utilisateur de plus, il affiche les histoires créer par l’utilisateur avec 2 boutons : un pour supprimer l’autre modifier. Elle redirige tous les utilisateurs non connectés vers la page login.php.

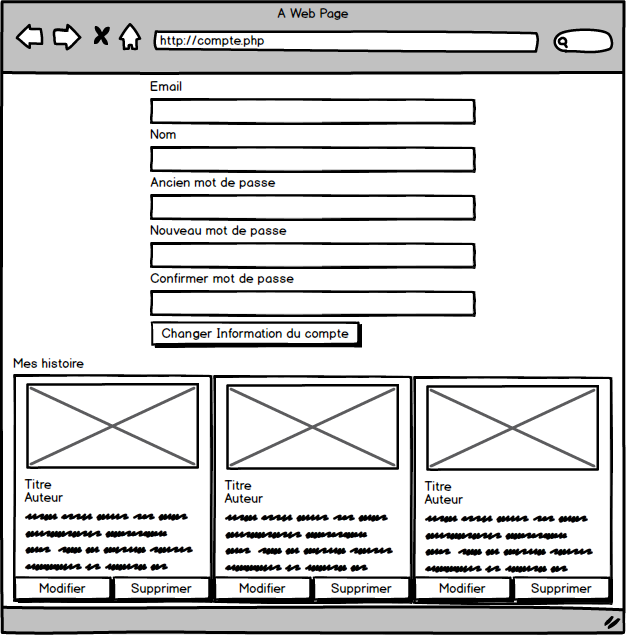


Figure 8 : Page compte.php

* connexion.php : cette page permet d’afficher les champs à l’utilisateur pour se connecter. Elle redirige l’utilisateur vers la page index.php s’il est déjà connecté.

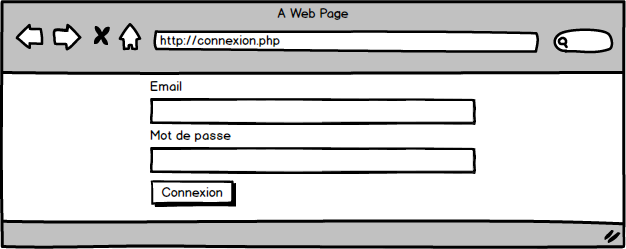


Figure 9 : Page connexion.php

* creerCompte.php : Comme pour la page connexion.php si l’utilisateur est déjà connecté alors la page le redirige vers index.php. Elle permet aussi d’afficher à l’utilisateur les champs pour créer un nouveau compte.

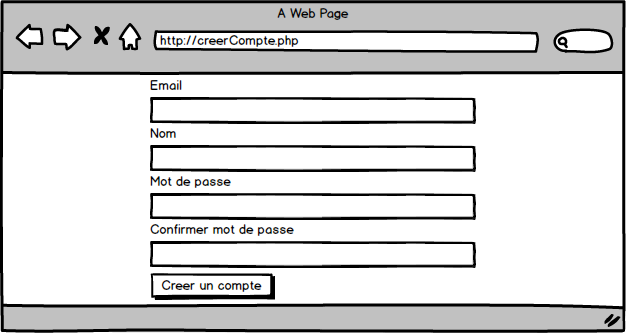


Figure 10 : Page creerCompte.php

* deconnecter.php : cette page n’affiche rien, mais enlève toutes les valeurs de la variable $\_SESSION et redirige l’utilisateur vers l’index.
* histoire.php : cette page affiche une histoire à l’utilisateur avec en plus de la moyenne de l’histoire et la moyenne de l’utilisateur, 4 listes déroulantes avec chacun d’eux les valeurs de 1-5 de pour chaque note pour noter l’histoire et si l’utilisateur a connecté un bouton pour rajouter l’histoire a ses favoris / enlever des ses favoris.

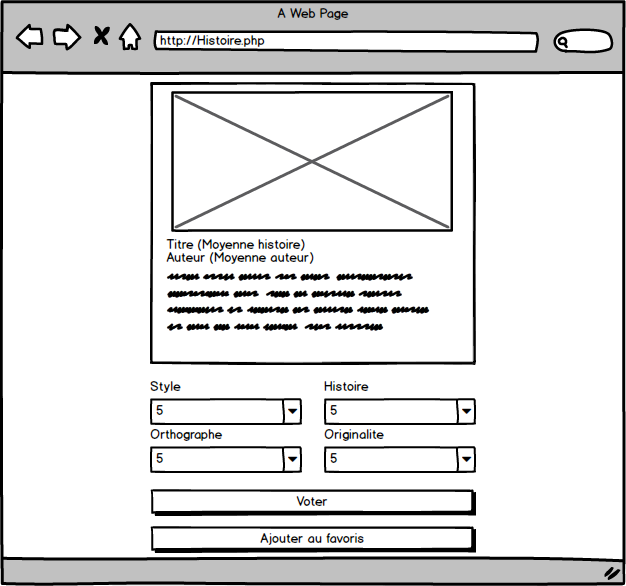


Figure 11 : Page histoire.php

* Index.php : cette page affiche les histoires que le contrôleur lui envoyée trier soit par date de création (ordre décroissant) soit pas par note (ordre décroissant) et si l’utilisateur est connecté on affiche les histoires favorites.

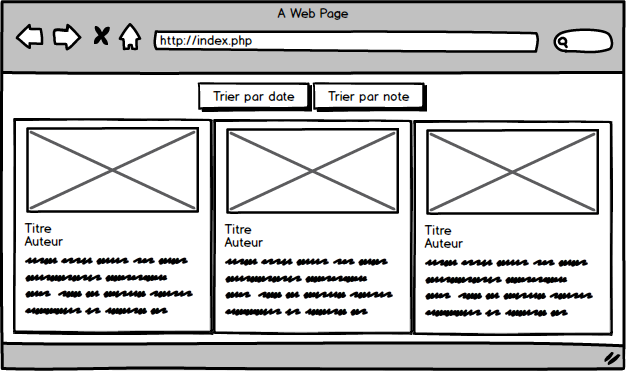


Figure 12 : Page index.php

* modifierHistoire.php : cette page permet d’afficher les champs pour modifier une histoire ou même créer une histoire.

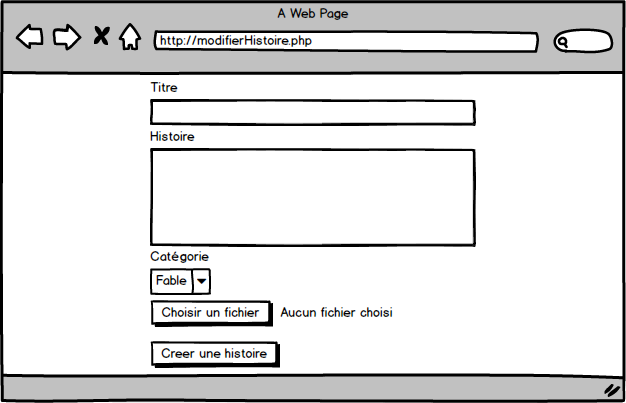


Figure 13 : Page modifierHistoire.php

* myCSS.css : CSS globale de l’application pour gérer l’affichage des images de l’histoire
* navbar.php : ce fichier est inclus dans toute les vus et permet gérer la barre de navigation et d’afficher différents liens dépendants de si utilisateur et connecter ou non.

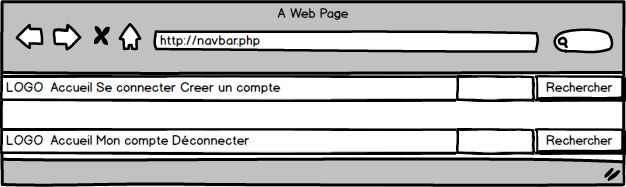


Figure 14 : Page navbar.php

* rechercher.php : cette page affiche le résultat d’une recherche avec toutes les histoires ayant le même titre que la recherche et toutes les histoires d’un utilisateur qui a le même nom que la recherche

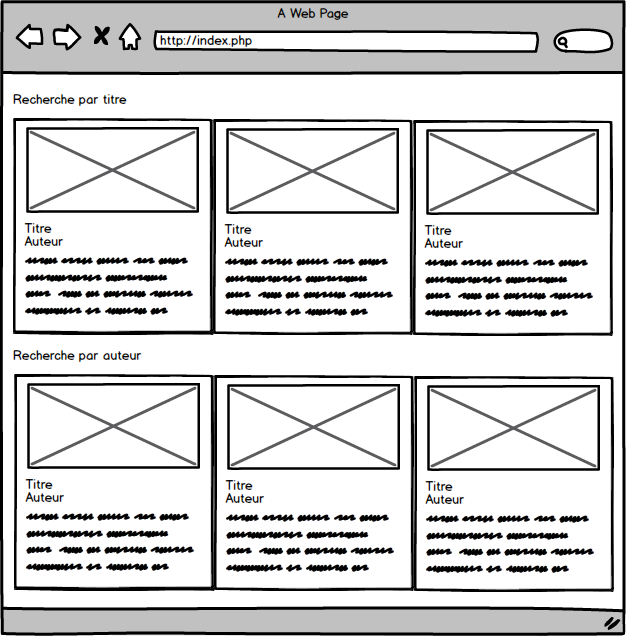


Figure 15 : Page index.php

* supprimer.php : permets d’afficher deux boutons oui / non pour supprimer une histoire

### TalesOfTheTavern/Vue/bootstrap-4.3.1-dist

Dossier contenant les fichiers nécessaires à Bootstrap.

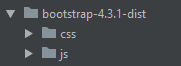


Figure 16 : Architecture Bootstrap

### TalesOfTheTavern/Vue/Img

Ce dossier contient toutes les images.



Figure 17 : Architecture Img

### TalesOfTheTavern/Controleur

Ce dossier contient les diffèrent contrôleur de chacune des vus en plus d’un contrôleur global.

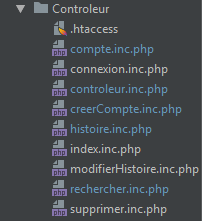


Figure 18 : Architecture Controleur

* .htaccess : permets d’éviter à des utilisateurs de pouvoir rentrer dans ce dossier sur le site web.
* controleur.inc.php : Ceci est le contrôleur principal et donc toutes les requêtes qui demande des informations à la base de données sont ici de plus il contient les fonctions récurrentes qui sont utilisées sur plusieurs pages
* compte.inc.php : d’abord une requête SQL est faite avec l’email de l’utilisateur qui est stocké dans $\_SESSION [utilisateur] si aucun utilisateur n’existe alors on le redirige sur la page login.php. Sinon, on affiche le nom et l’email dans leurs champs respectifs. S’il y a des valeurs en POST et que le SHA256 du champ « ancien mot de passe » est égal au mot de passe stocker dans la base. Alors, on modifie le nom et l’email (en vérifiant que cet email n’est pas déjà utilisé), de plus, si les champs « nouveau mot de passe » et « confirmer nouveau mot de passe » ne sont pas nuls et respecte la condition de 8 caractères de long minimum et contient au moins une lettre et un chiffre. Alors on modifie aussi le mot de passe de l’utilisateur. On appelle aussi la fonction AfficherHistoire() qui est dans le « controleur.inc.php » et en mettant la valeur vraie au paramètre « afficherBouton » cela permet d’afficher les boutons de modification(redirection vers modifierHistoire.php avec en GET l’id de l’histoire) et de suppression (redirection vers supprimer.php avec en GET l’id de l’histoire).
* connexion.inc.php : S’il y a des valeurs en POST alors on vérifie que l’email que l’utilisateur nous a fourni existe bien dans notre base et ensuite on vérifie que le SHA256 de l’utilisateur est égal au mot de passe stocker dans la base si c’est le cas, on stocke l’email dans $\_SESSION[utilisateur] sinon on affiche un message d’erreur.
* creerCompte.inc.php : S’il y a des valeurs en POST alors on vérifie que l’email que l’utilisateur nous a fourni n’existe pas dans notre base. Sinon, on affiche un message d’erreur. Ensuite, on vérifie que le mot de passe est égal au mot de passe de confirmation et qu’il fasse 8 caractères de long minimum et qu’il contient au moins une lettre et un chiffre. Si toutes ses conditions sont réunies alors on stocke ces informations dans la base et comme pour la connexion on stocke l’email dans $\_SESSION[utilisateur].
* histoire.inc.php : Si la variable ID en GET n’est pas égale à l’id d’une histoire dans la base de données alors on redirige l’utilisateur vers la page index.php. Sinon, on vérifie que $\_SESSION[utilisateur] existe et si c’est le cas on affiche le bouton « ajouter/supprimer aux favoris » si utilisateur à cette histoire en favoris ou non. Enfin, s’il y a des valeurs en POST alors et que $\_POST[favoris] n’existe pas, alors on insère les valeurs des énum dans la base de données or, $\_POST[favoris] existe c’est-à-dire que l’utilisateur veut rajouter ou supprimer aux favoris.
* Index.inc.php : Si la valeur de la variable ordre en GET n’est pas égale à la moyenne ou n’existe pas, alors on récupère les histoires et les histoires favorites si l’utilisateur est connecté, trier par la date avec un ordre décroissant. Sinon on affiche les histoires, et les favoris, trier par moyenne aussi avec un ordre décroissant.
* modifierHistoire.php : si id en GET n’existe pas alors on affiche la vue avec aucune information stocker dans les champs et dès qu’on appuie sur le bouton ça ajoute une nouvelle histoire. Sinon, on vérifie que l’histoire existe bien et que c’est bien l’utilisateur qui a créé cette histoire (si ça n’est pas le cas, alors on le redirige vers compte.php) alors, dès qu’il appuie sur le bouton on modifie son histoire et s’il n’a pas rajouté d’image alors on laisse l’ancienne image.
* Rechercher.inc.php : Si en GET la variable recherche existe alors on demande au contrôleur de nous retourner toutes les histoires contenant se texte et si un utilisateur a se nom de retourner toutes ses histoires. Sinon on redirige l’utilisateur vers index.php
* Supprimer.inc.php : si id en GET n’existe pas ou que ce n’est pas cet utilisateur qui a créé cette histoire, alors on redirige l’utilisateur vers compte.php. Sinon, alors dès qu’il appuie sur le bouton oui on le redirige vers la page compte.php en supprimant les informations de l’histoire. Si l’utilisateur appuie sur non alors on le redirige juste vers compte.php.

### TalesOfTheTavern/Model

* htaccess : permets d’éviter à des utilisateurs de pouvoir rentrer dans ce dossier sur le site web.
* connexionDB.php: ce fichier contient la fonction qui permet de se connecter à la base de données.
* constanteDB.php : ce fichier contient les constantes pour se connecter à la base de données username, password, host**,** name.
* fonctionDB.php : ce fichier contient toutes les fonctions que le contrôleur.inc.php va appeler et ce fichier va retourner les valeurs depuis la base

## Analyse de plusieurs fonctionnalités principales

### Découpage MVC

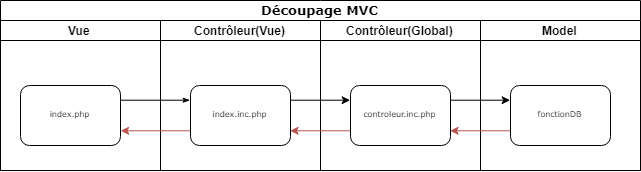


Figure 19 : découpage MVC avec comme page d'exemple index

## Moyenne utilisateur

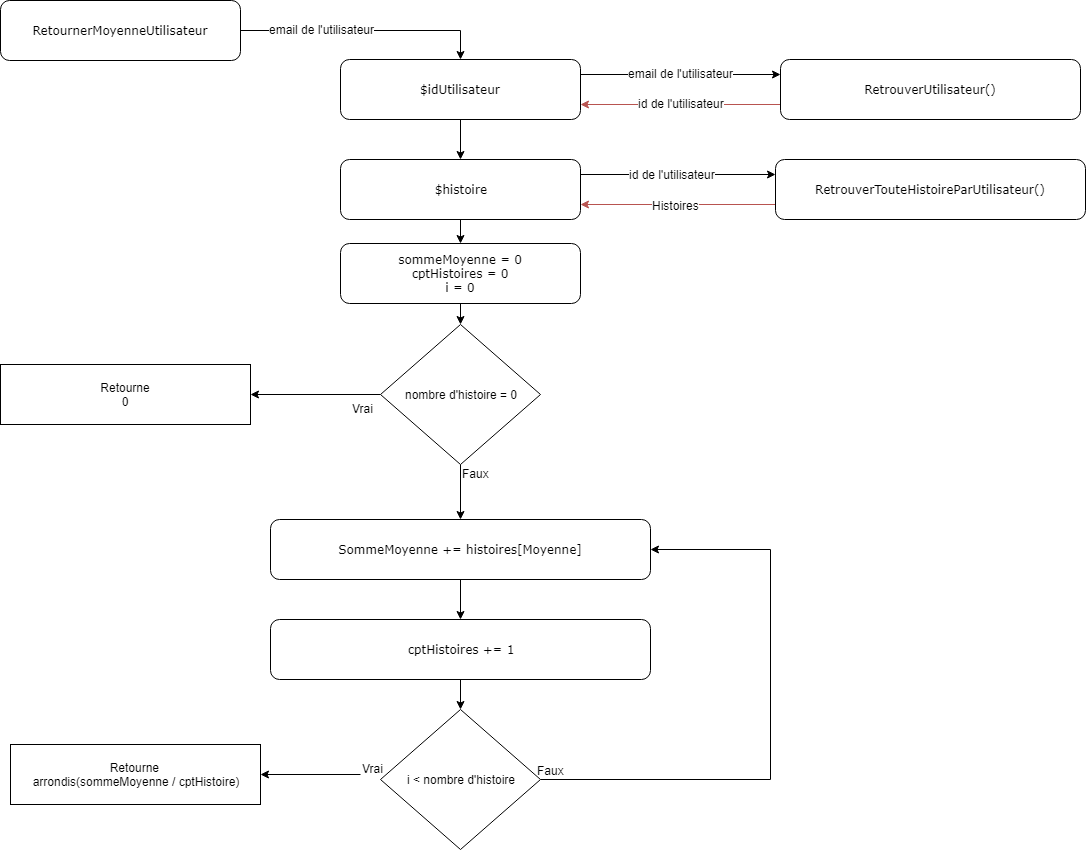


Figure 20 : logique moyenne d'un utilisateur

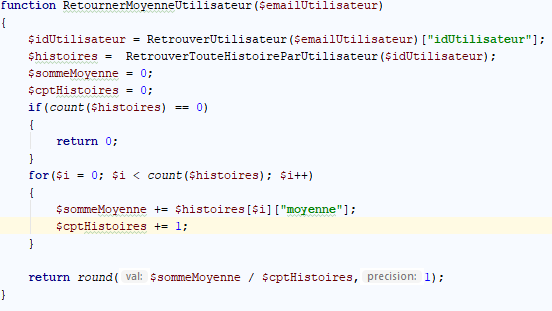


Figure 21 : code moyenne d'un utilisateur

# Framework

Pour Tales of the taverne j’ai utilisé deux librairies.

## Bootstrap



Figure 22 : Logo Bootstrap

C’est une boite à outils qui permet de faire des sites plus beaux et responsive beaucoup plus facilement.

Il faut installer la distribution et rajouter 4 lignes de code pour pouvoir faire fonctionner Bootstrap. Ensuite, il suffit juste mettre une ou plusieurs classes à une balise et Bootstrap s’occupe du reste.

<https://getbootstrap.com/>

V. 4.3.1

## Font Awesome



Figure 23 : Logo Font Awesome

C’est une bibliothèque d’icône / logo.

Pour l’utiliser il suffit de rajouter une ligne code pour relier au CSS en ensuite il suffit de rechercher sur leur site ce que nous voulons et de copier la ligne de code.

<https://fontawesome.com/>

V. 5.8.1

# Test et plan de test

Tableau 3 : Test du 02.04.19

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Critère | ID | Test | Résultat attendu | Résultat produit | Résultat | Date de validation |
| Fonctionnalité | | | | | | |
| A14 | F1 | Création compte + connexion | Un utilisateur peut créer un compte et s'y connecter | Il y a une page connexion.php qui permet de se connecter et une page creerCompte.php | KO |  |
| A15 | F2 | Redirection anonyme | Un utilisateur anonyme ne peut pas accéder à la page compte.php | un utilisateur anonyme et redirigé vers la page index.php | KO |  |
| F3 | Gestion informations personnelles | Un utilisateur authentifié peut modifier ses informations | Un utilisateur authentifier peut changer ses informations dans la page compte.php | KO |  |
| F4 | Gestion histoire | Un utilisateur authentifié peut ajouter / supprimer / modifier une histoire | Un utilisateur authentifier peut ajouter / modifier / supprimer ses histoires dans la page compte.php | KO |  |
| F5 | Persistance BD | Le CRUD de l'histoire et des informations personnelles sont bien faits dans la BD et peuvent être vérifiés par l'administrateur | Les informations sont bien changées dans la BD | KO |  |
| A16 | F6 | Trie page accueil | Un utilisateur anonyme peut trier la page d'accueil par note ou par moyenne (ordre décroissant) | Un utilisateur peut trier la page d'accueil avec 2 boutons | KO |  |
| A17 | F7 | non-affichage anonyme | Un utilisateur anonyme ne peut pas accéder à la fonctionnalité favoris | La fonctionnalité ne s'affiche pas pour l'utilisateur anonyme | KO |  |
| F8 | Gestion favoris | Un utilisateur authentifié peut ajouter / supprimer des favoris | Un utilisateur authentifier peut ajouter / supprimer ses favoris | KO |  |
| F9 | Persistance BD | L'ajout et la suppression des favoris sont bien faits dans la BD et peuvent être vérifiés par l'administrateur | Les informations sont bien changées dans la BD | KO |  |
| A18 | F10 | Vote | un utilisateur anonyme peut évaluer une histoire sur 4 critères | un utilisateur anonyme peut évaluer une histoire sur 4 critères | KO |  |
| F11 | Persistance BD | une évaluation est bien ajoutée à la base de données et peut être vérifiée par l'administrateur | Les informations sont bien changées dans la BD | KO |  |
| A19 | F12 | Calcule moyenne histoire | l'administrateur peut afficher les moyennes de chacun des critères et la moyenne globale de l'histoire | l'administrateur peut afficher les moyennes de chacun des critères et la moyenne globale grâce à une requête SQL | KO |  |
| F13 | Calcule moyenne auteur | l'administrateur peut afficher la moyenne d'un utilisateur | l'administrateur peut afficher la moyenne d'un auteur grâce à une requête SQL | KO |  |
| A20 | F14 | Recherche | Un utilisateur anonyme peut rechercher une histoire par le titre ou l'auteur. | l'utilisateur peut rechercher une histoire par titre ou par auteur (affiche toutes les histoires de l'auteur) | KO |  |

Tableau 4 : Test du 15.04.19

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Critère | ID | Test | Résultat attendu | Résultat produit | Résultat | Date de validation |
| Fonctionnalité | | | | | | |
| A14 | F1 | Création compte + connexion | Un utilisateur peut créer un compte et s'y connecter | Il y a une page connexion.php qui permet de se connecter et une page creerCompte.php | OK | 04.04.19 |
| A15 | F2 | Redirection anonyme | Un utilisateur anonyme ne peut pas accéder à la page compte.php | un utilisateur anonyme et redirigé vers la page index.php | OK | 08.04.19 |
| F3 | Gestion informations personnelles | Un utilisateur authentifié peut modifier ses informations | Un utilisateur authentifier peut changer ses informations dans la page compte.php | OK | 09.04.19 |
| F4 | Gestion histoire | Un utilisateur authentifié peut ajouter / supprimer / modifier une histoire | Un utilisateur authentifier peut ajouter / modifier / supprimer ses histoires dans la page compte.php | OK | 09.04.19 |
| F5 | Persistance BD | Le CRUD de l'histoire et des informations personnelles sont bien faits dans la BD et peuvent être vérifiés par l'administrateur | Les informations sont bien changées dans la BD | OK | 09.04.19 |
| A16 | F6 | Trie page accueil | Un utilisateur anonyme peut trier la page d'accueil par note ou par moyenne (ordre décroissant) | Un utilisateur peut trier la page d'accueil avec 2 boutons | OK | 10.04.19 |
| A17 | F7 | non-affichage anonyme | Un utilisateur anonyme ne peut pas accéder à la fonctionnalité favoris | La fonctionnalité ne s'affiche pas pour l'utilisateur anonyme | OK | 10.04.19 |
| F8 | Gestion favoris | Un utilisateur authentifié peut ajouter / supprimer des favoris | Un utilisateur authentifier peut ajouter / supprimer ses favoris | OK | 10.04.19 |
| F9 | Persistance BD | L'ajout et la suppression des favoris sont bien faits dans la BD et peuvent être vérifiés par l'administrateur | Les informations sont bien changées dans la BD | OK | 10.04.19 |
| A18 | F10 | Vote | un utilisateur anonyme peut évaluer une histoire sur 4 critères | un utilisateur anonyme peut évaluer une histoire sur 4 critères | OK | 11.04.19 |
| F11 | Persistance BD | une évaluation est bien ajoutée à la base de données et peut être vérifiée par l'administrateur | Les informations sont bien changées dans la BD | OK | 11.04.19 |
| A19 | F12 | Calcule moyenne histoire | l'administrateur peut afficher les moyennes de chacun des critères et la moyenne globale de l'histoire | l'administrateur peut afficher les moyennes de chacun des critères et la moyenne globale grâce à une requête SQL | OK | 11.04.19 |
| F13 | Calcule moyenne auteur | l'administrateur peut afficher la moyenne d'un utilisateur | l'administrateur peut afficher la moyenne d'un auteur grâce à une requête SQL | OK | 11.04.19 |
| A20 | F14 | Recherche | Un utilisateur anonyme peut rechercher une histoire par le titre ou l'auteur. | l'utilisateur peut rechercher une histoire par titre ou par auteur (affiche toutes les histoires de l'auteur) | OK | 12.04.19 |

# Conclusion

Je suis très content du travail que j’ai fourni et de l’état actuel de mon projet. Il respecte le cahier des charges et j’ai beaucoup apprécié travailler dessus. Je trouve que ma planification était juste, or il est vrai que pour la tâche « Un utilisateur authentifié peut gérer ses histoires » j’ai pris plus de temps à cause de certaines erreurs, mais j’ai pu le rattraper ce retard. Ceci m’a appris que lorsque je planifie quelque chose il faut toujours un temps supplémentaire en cas de problème.

## Difficultés rencontrées

Une des difficultés que j’ai rencontrées était la vérification de l’email, j’ai d’abord voulu le faire en Regex et je me suis vite rendu compte que cela allait être plus compliqué que prévu, c’est alors que dans une de mes recherches pour trouver une autre solution, j’ai découvert un filtre en PHP qui permet de vérifier si une chaine de caractère est un émail ou non.

## Amélioration possible

* Meilleure gestion des différentes tailles de l’image lors de l’affichage
* Supprimer une image lors de la modification d’une image si elle est modifiée
* Correction Scroll verticale lors de l’affiche d’image
* Changer les vérifications du mot de passe en JavaScript plutôt que PHP
* Système d’envoi d’email lors de la modification de l’email

## Remerciements

Je remercie le CFPT et, tout particulièrement, Mme Mota pour m’avoir suivi durant toute la durée du TPI et de m’avoir donné des conseils pour la réalisation.

# Bibliographie

Manuel PHP : <https://www.php.net/manual/fr/index.php>

OpenClassroom : <https://openclassrooms.com/>

Stack Overflow : <https://stackoverflow.com/>

Github : <https://github.com/zabraf/TalesOfTheTavern>

SitePoint : <https://www.sitepoint.com/community/t/check-whether-string-contains-numbers/5953>

Pixilart : <https://www.pixilart.com>

Bootstrap : <https://getbootstrap.com/>

Font awesome : <https://fontawesome.com/>

Unsplash : [www.unsplash.com](http://www.unsplash.com)

# Annexes

## Glossaire

|  |  |
| --- | --- |
| NOM | Signification |
| Tales of the tavern | Les histoires de la taverne en anglais |
| PHP | PHP: Hypertext Preprocessor : langage de programmation pour faire des pages web dynamiques |
| MVC | Model-Vue-Contrôleur |
| MLD | Model Logique de Données |
| MCD | Model Conceptuel de Données |
| SQL | Structured Query Language : langage de programmation de base de données |
| BD | Base de données |
| GANTT | Outil de gestion de projet (planification) |

## Énoncer

## Planification

## Journal de bord

## Résumé TPI

## Code source