2024-2025

TASKSTEP : AUDIT D'OPTIMISATION DE QUALITE LOGICIELLE

1PEUDINSPI :

- Tristan DAL MOLIN
- Moulay-Wassim ALAOUI
- Jules DUTRION
- Matteo DE MARCO
- Wassim DIOURI

BUT2 Informatique

IUT DIJON AUXERRE NEVERS
UNIVERSITÉ DE BOURGOGNE
Révélateur de talents depuis 1968

UNIVERSITÉ DE BOURGOGNE

I. Ancienne conception du projet

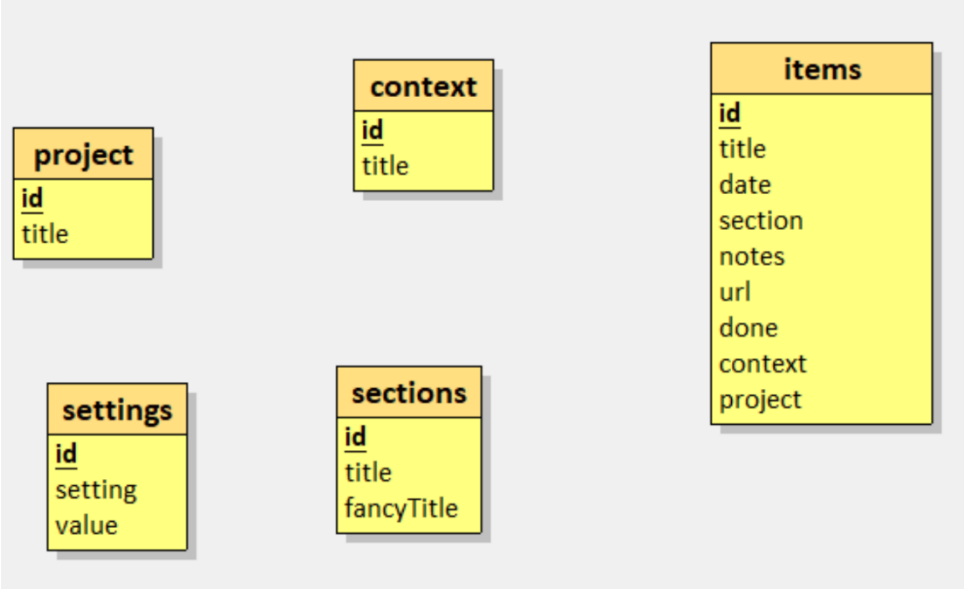


Figure 1 : Modèle de Conception de Données du projet

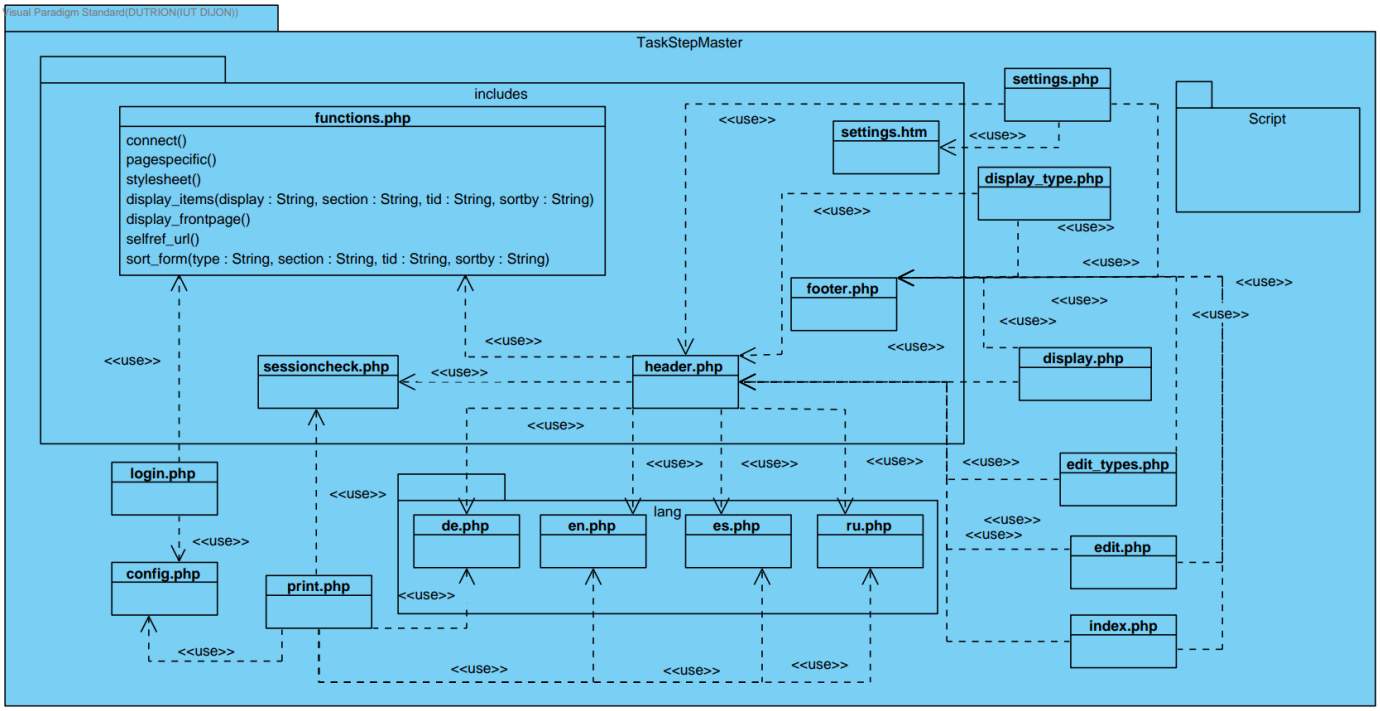


Figure 2 : Conception de l'application (UML)

II. Nouvelle conception du projet

Afin de remanier l'application de façon SOLID et optimisée, nous avons refait la structure du projet de 0 par Programmation Orientée Objets en PHP, utilisant notamment une structure MVC :

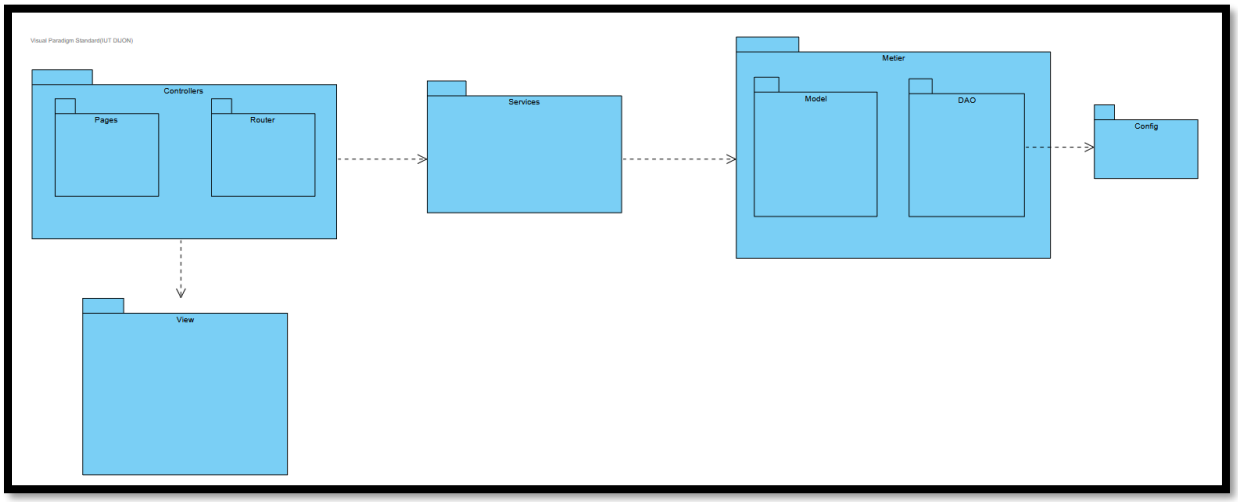


Figure 3 : Diagramme des packages

III. Audit d'optimisation de qualité logicielle

L'audit qui suit contient les résultats de l'audit principal ainsi que ses corrections et optimisations en dernière colonne.

La notation des résultats est la suivante :

- Le point est une bonne pratique identifiée (point très bien respecté) : BP
- Le point est conforme (c'est respecté, éventuellement avec quelques limites) : CF
- Le point est non conforme (c'est insuffisamment respecté) : NC
- Le point est critique (ce n'est pas du tout respecté, c'est totalement absent…) : CR

Critères d'audit	Outils utilisés	Descriptions des vérifications effectuées (tests, conditions de test, etc.)	Résultats de l'audit (constats)	Optimisation
A – Documentation technique				
A.1 – Documentation externe				
A.1.1 Document indiquant la plateforme d'exécution (Qu'installer sur une VM pour exécuter l'appli)	Visual Studio Code	Lecture du fichier README.htm pour analyser les étapes d'installation locale et vérifier les composants nécessaires (AMP).	BP – L'application doit être exécutée sur un serveur local type XAMPP contenant Apache, MySQL et PHP et tout est indiqué en README.	Rien à modifier
A.1.2 Document indiquant la plateforme de développement	Visual Studio Code	Lecture du README.md pour vérifier la plateforme de développement	BP – Le projet est développé en PHP. Aucun environnement spécifique n'est imposé, tout est indiqué en README	Rien à modifier
A.1.3 Document décrivant l'architecture globale de l'application	Visual Studio Code	Vérification que l'architecture du projet est présente dans sa documentation (UML, conception, …)	N/A – Aucune architecture, non applicable	Un fichier .vpp contient la conception complète du projet (cf la nouvelle conception du projet)
A.1.4 Document décrivant les dépendances aux outils tiers de l'application	Visual Studio Code	Lecture du README.md pour vérifier les dépendances aux outils tiers de l'application (API extérieures, logiciels à installer en plus de l'application de base)	BP – Présence des dépendances du projet en README. La plateforme d'exécution nécessite PHP ≥ 7.0, MariaDB et un serveur local pour exécuter le projet	Aucune dépendance supplémentaire

A.1.5 Cahier des charges ou document décrivant les fonctionnalités de l'application	Visual Studio Code	Lecture du README.md pour vérifier si les fonctionnalités y sont décrites. (Diagrammes de séquences, descriptions des fonctionnalités)	CF – Les fonctionnalités sont listées dans le README : organisation des tâches, filtres contextuels/projets, affichage auto des tâches du jour, impression, surlignage des tâches urgentes ou en retard, multilingue (Anglais, Russe, Allemand, Espagnol).	Un UML du projet entier a pu être réalisé, comprenant notamment un diagramme de cas d'usage de l'application

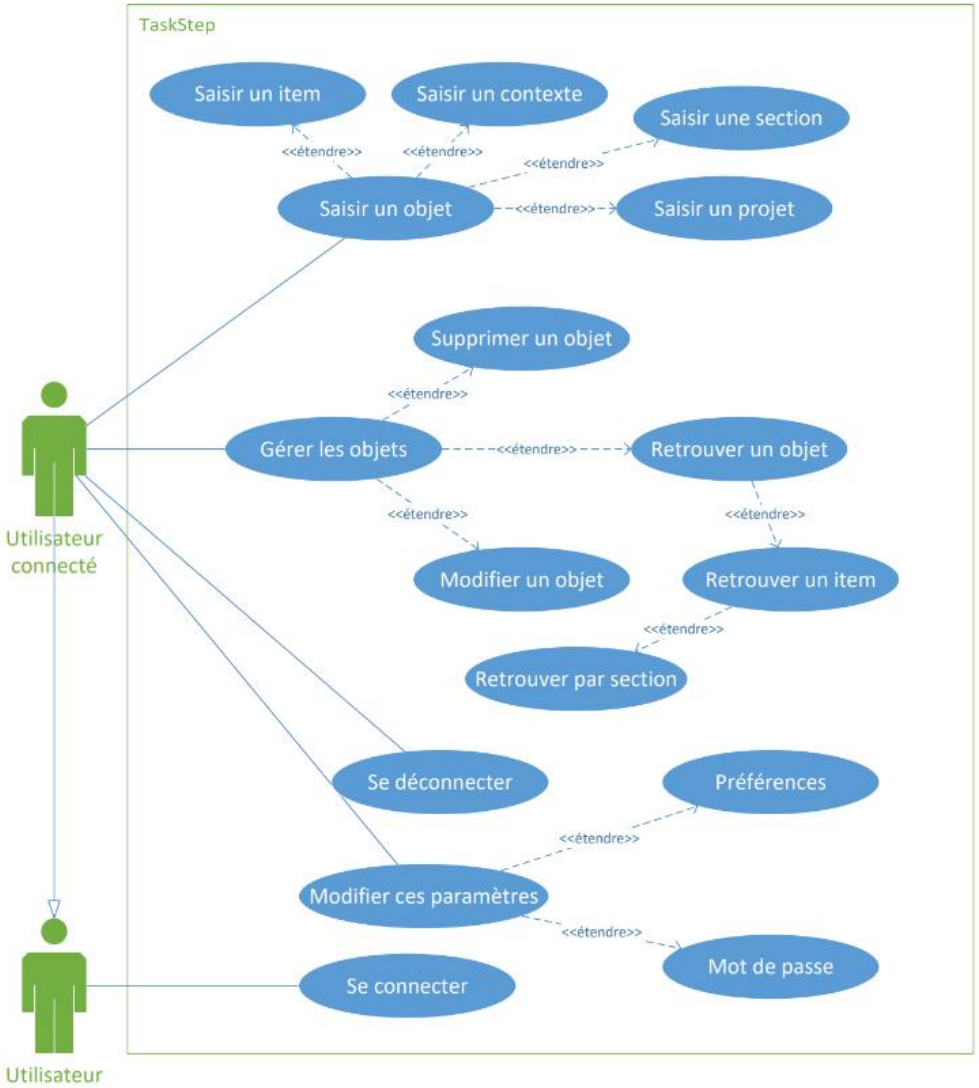


Figure 4 : Diagramme de cas d'usage

Critères d'audit	Outils utilisés	Descriptions des vérifications effectuées (tests, conditions de test, etc.)	Résultats de l'audit (constats)	Optimisation
A – Documentation technique				
A.2 – Documentation interne				
A.2.1 Documentation in-situ du code présente (Code documenté)	Visual Studio Code	Analyse du code existants des différents fichiers présents dans le projet.	CR – Il semble qu'il n'y ait vraiment aucune documentation dans tout le projet. Seulement des commentaires rarement.	Documentation, commentaire, mis en place sur tout le projet (en anglais)
A.2.2 Documentation du code cohérente avec le code et à jour	N/A	Vérification que la documentation a du sens et a une version, que les paramètres y soient ajoutés si ajoutés à une version du site	N/A (Pas de documentation)	
A.2.3 Documentation du code cohérente avec les spécifications fonctionnelles (cf A.1.5., si ça respecte les fonctionnalités décrites)	N/A	Présence de documentation cohérente avec les spécifications	N/A (Pas de documentation)	
A.2.4 Documentation du code claire, lisible, dans la langue souhaitée	N/A	Documentation claire, lisible et dans une même langue	N/A (Pas de documentation)	

Controllers \ Pages \ Home

ControllerHome

in package Application

ControllerHome.php : 10

The controller for the home page

Table of Contents

Properties

P $page : HomePage

Methods

M __construct() : mixed

The class constructor

M index() : void

Method showing the home page

Logic \ DAO \ Interface

I_TaskDAO

in Application

I_TaskDAO.php : 14

DAO interface for the task table

Table of Contents

Methods

- AddTask() : void
  Methode to add a task to the database
- DeleteTask() : void
  Method to delete a task in the database
- GetTask() : Task
  Method to get a task from the database
- ListTask() : array<string|int, Task>
  Method to list the tasks
- ListTaskByContext() : array<string|int, Task>
  Method to list the tasks by section
- ListTaskByProject() : array<string|int, Task>
  Method to list the tasks by project
- ListTaskBySection() : array<string|int, Task>
  Method to list the tasks by section
- TaskDone() : void
  Method to set a task as done
- UpdateTask() : void

Routes

extends RoutesAbstract
in package Application

Routes.php : 57

Class listing all the roads of the project

Table of Contents

Properties

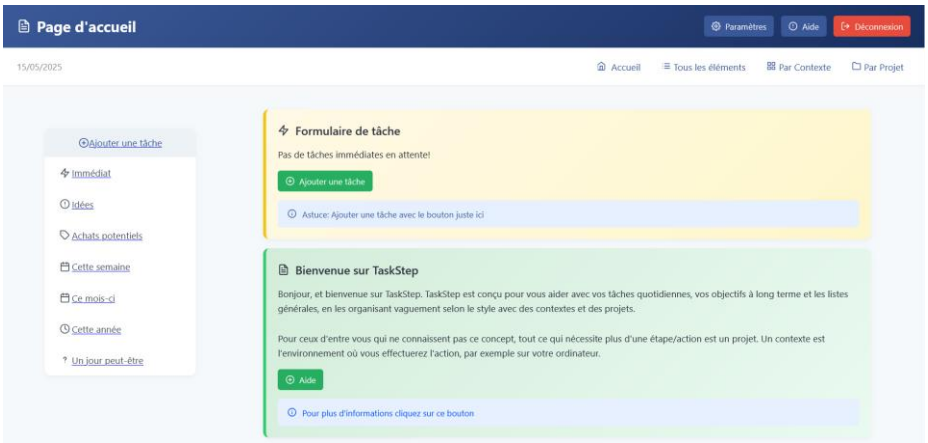
- $container : ContainerInterface
- $ctrlList : array<string|int, mixed>
- $routeList : array<string|int, mixed>

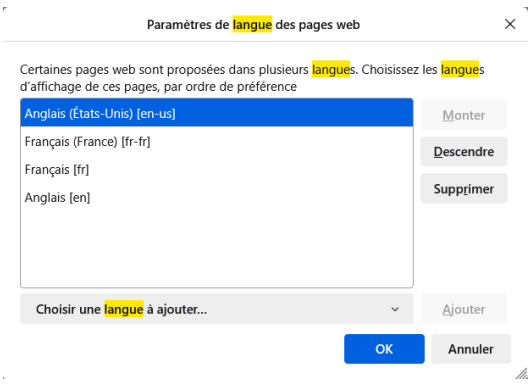
Methods

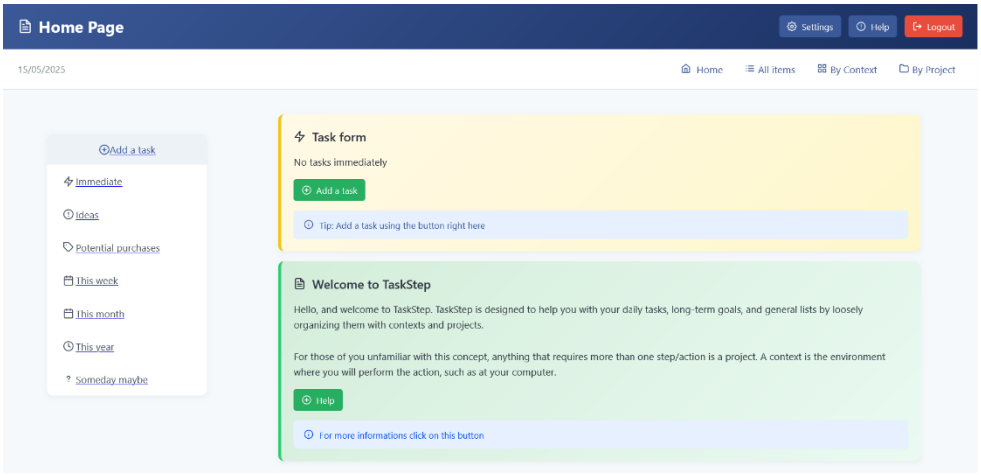
- __construct() : mixed
  The class constructor
- createControllerList() : void
  Method to initialize the list of controllers
- createRouteList() : void
  Method to initialize the list of routes
- getConnexionRoute() : string
  Method to obtain the name of the connexion page
- getDefaultRoute() : string
  The default route getter for the app
- getRoute() : Route
  Method returning the asked route or the default one

Critères d'audit	Outils utilisés	Descriptions des vérifications effectuées (tests, conditions de test, etc.)	Résultats de l'audit (constats)	Optimisation
B – Tests de l'application				
B.1 – Tests fonctionnels				
B.1.1 Les cas d'usage de l'application sont tous implémentés (Respect des diagrammes d'usage en conception)	N/A	Vérification que les cas d'usage sont tous respectés	N/A -- Pas de diagrammes de conceptions	/
B.1.2 Les spécifications fonctionnelles sont respectées	Firefox/ Edge	Vérifications que les fonctionnalités de la documentation sont bien respectées	CF – La plupart sont conformes, excepté le changement de langue du navigateur (changement de langue possible via le fichier config)	Changement de langue implémenté selon la langue du navigateur directement

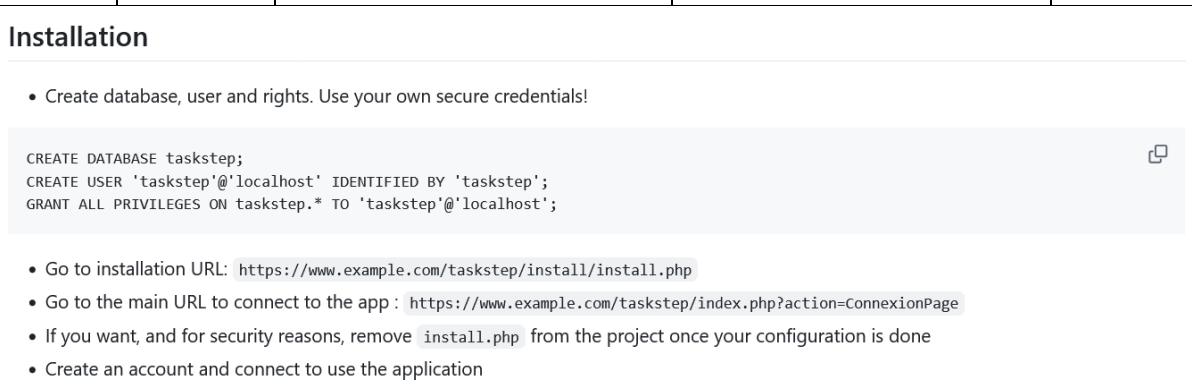
Exemple de changement de langue :







B.1.3 L'application fonctionne sur les plateformes d'exécution spécifiées	Oracle vm virtualbox	Vérifier que l'application fonctionne sous machine virtuelle en suivant le guide d'installation et d'utilisation.	CF – Tutoriel de mise en place du serveur fonctionnel, à mettre à jour avec la mise à jour du projet	Fichier README mis à jour avec l'optimisation du logiciel



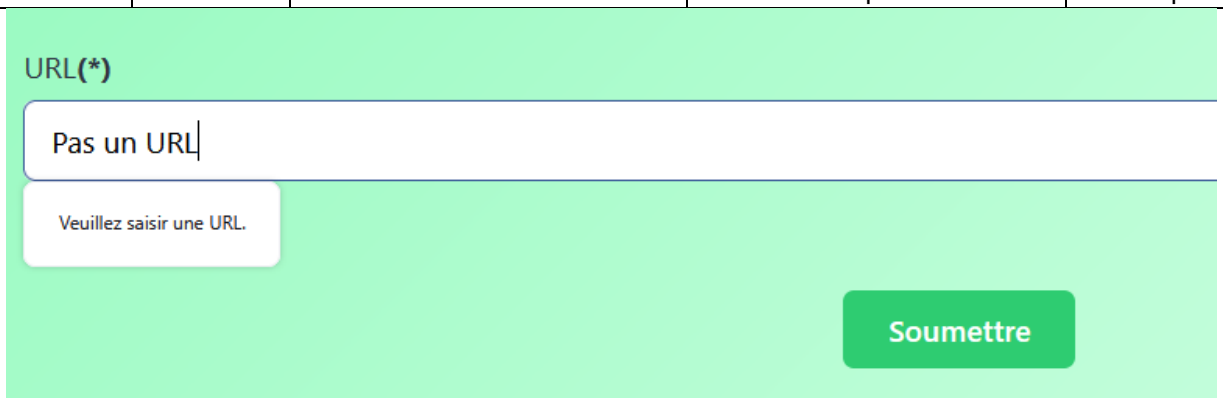
Installation

- Create database, user and rights. Use your own secure credentials!

```
CREATE DATABASE taskstep;
CREATE USER 'taskstep'@'localhost' IDENTIFIED BY 'taskstep';
GRANT ALL PRIVILEGES ON taskstep.* TO 'taskstep'@'localhost';
```

- Go to installation URL: `https://www.example.com/taskstep/install/install.php`
- Go to the main URL to connect to the app : `https://www.example.com/taskstep/index.php?action=ConnexionPage`
- If you want, and for security reasons, remove `install.php` from the project once your configuration is done
- Create an account and connect to use the application

B.1.4 L'application est robuste aux cas particuliers	Firefox	Vérification qu'aucune mauvaise saisie n'est possible (lettres à la place d'un nombre par exemple)	NC – Insertion possibles avec des champs nécessaires vides, même nom de contexte et de projet possibles, une date peut être entrée en tant que texte et non que date	Formulaire mis à jour, empêchant d'entrer autre chose qu'un lien dans le champs URL ou autre chose qu'une date en date par exemple



Critères d'audit	Outils utilisés	Descriptions des vérifications effectuées (tests, conditions de test, etc.)	Résultats de l'audit (constats)	Optimisation
B – Tests de l'application				
B.2 – Tests unitaires				
B.2.1 Les tests unitaires sont présents et cohérents avec les spécifications	Visual Studio Code	Vérification que l'application a des tests unitaires cohérents pour les cas généraux et particuliers	NC – Pas de tests unitaires	Tests unitaires mis en place séparément sur le site web, vérifiant les interactions avec la base de données. Seuls les tests en couche DAO ont pu être réalisés, ils sont tous fonctionnels
B.2.2 Les tests unitaires passent tous	Visual Studio Code	Vérification que tous les tests unitaires passent	N/A -- Pas de tests unitaires	
B.2.3 Les tests unitaires sont indépendants les uns des autres	Visual Studio Code	Tests unitaires indépendants entre eux, simulation de couches inférieures, fakes DAO par exemple	N/A -- Pas de tests unitaires	
B.2.4 les cas particuliers (entrées invalides, effets de bord) sont testés	Visual Studio Code	Vérification que les cas particuliers sont testés	N/A -- Pas de tests unitaires	
B.2.5 la couverture des tests est correcte	Visual Studio Code	Couverture des tests correcte (pourcentage éventuel)	N/A -- Pas de tests unitaires	

Tests Unitaires

Unit test for Context

- Insert Test : Success
- ListContext Test : Success
- GetById Test : Success
- Update Test : Success
- Delete Test : Success

Unit test for Project

- Insert Test : Success
- GetById Test : Success
- Update Test : Success
- list all Test : Success
- Delete Test : Success

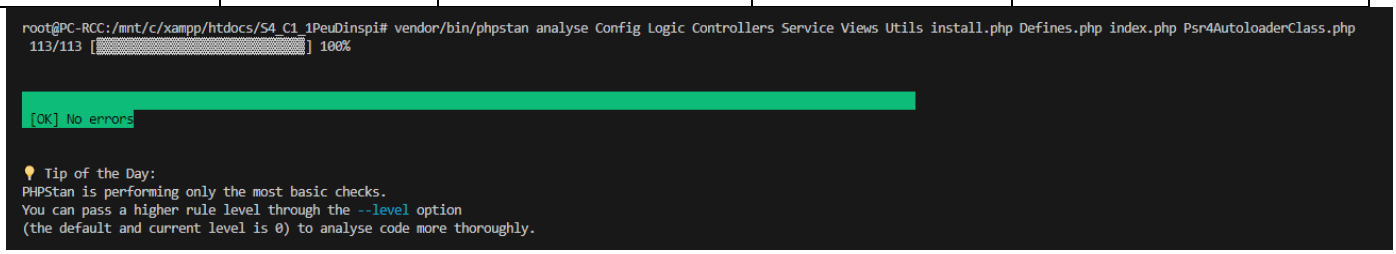
Unit test for the Task Table

- Insert Test : Success
- GetById Test : Success
- Update Test : Success
- Delete Test : Success
- List all Test : Success
- List by Context Test : Success
- List by Project Test : Success
- List by Section Test : Success

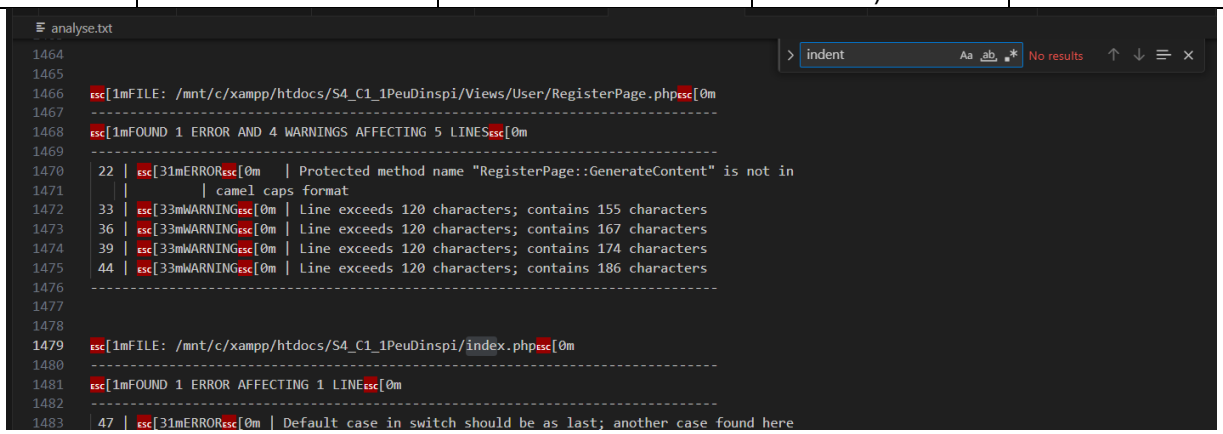
Unit test for Section

- TestAddSection : Success
- TestGetSection : Success
- TestUpdateSection : Success
- TestDeleteSection : Success
- TestListSection : Success

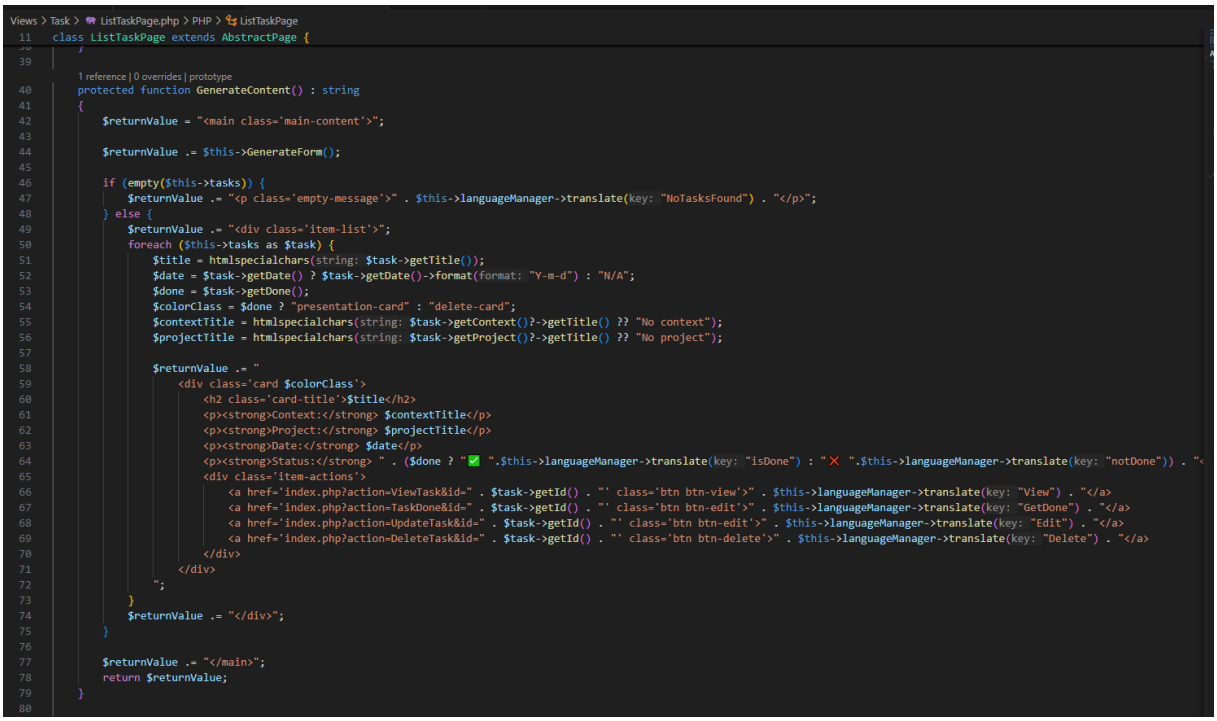
Critères d'audit	Outils utilisés	Descriptions des vérifications effectuées (tests, conditions de test, etc.)	Résultats de l'audit (constats)	Optimisation
C – Format du code source				
C.1 – Construction du code				
C.1.1 Lors de la construction du code (compilation) il n'y a pas d'avertissement (À modifier si c'est dangereux, peuvent être laissés de côté sinon)	« phpstan » sous Composer et Linux, Visual Studio Code	Commande « phpstan analyse » effectuée sur chaque classe PHP du projet	NC – Plus de 700 erreurs sont trouvées au total, dont beaucoup de méthodes appelées à des éléments	Après refonte du projet dans une version plus récente de PHP, le test sous phpstan ne renvoie désormais plus aucune erreur
C.1.2 Lors de la construction du code il n'y a pas d'erreurs	« phpstan » sous Composer et Linux, Visual Studio Code	Commande « phpstan analyse » effectuée sur chaque classe PHP du projet, sous une configuration nous permettant d'obtenir tous les résultats possibles	NC – Plus de 700 erreurs sont trouvées au total, beaucoup de non-typage, de méthodes appelées sur des éléments « mixed »	

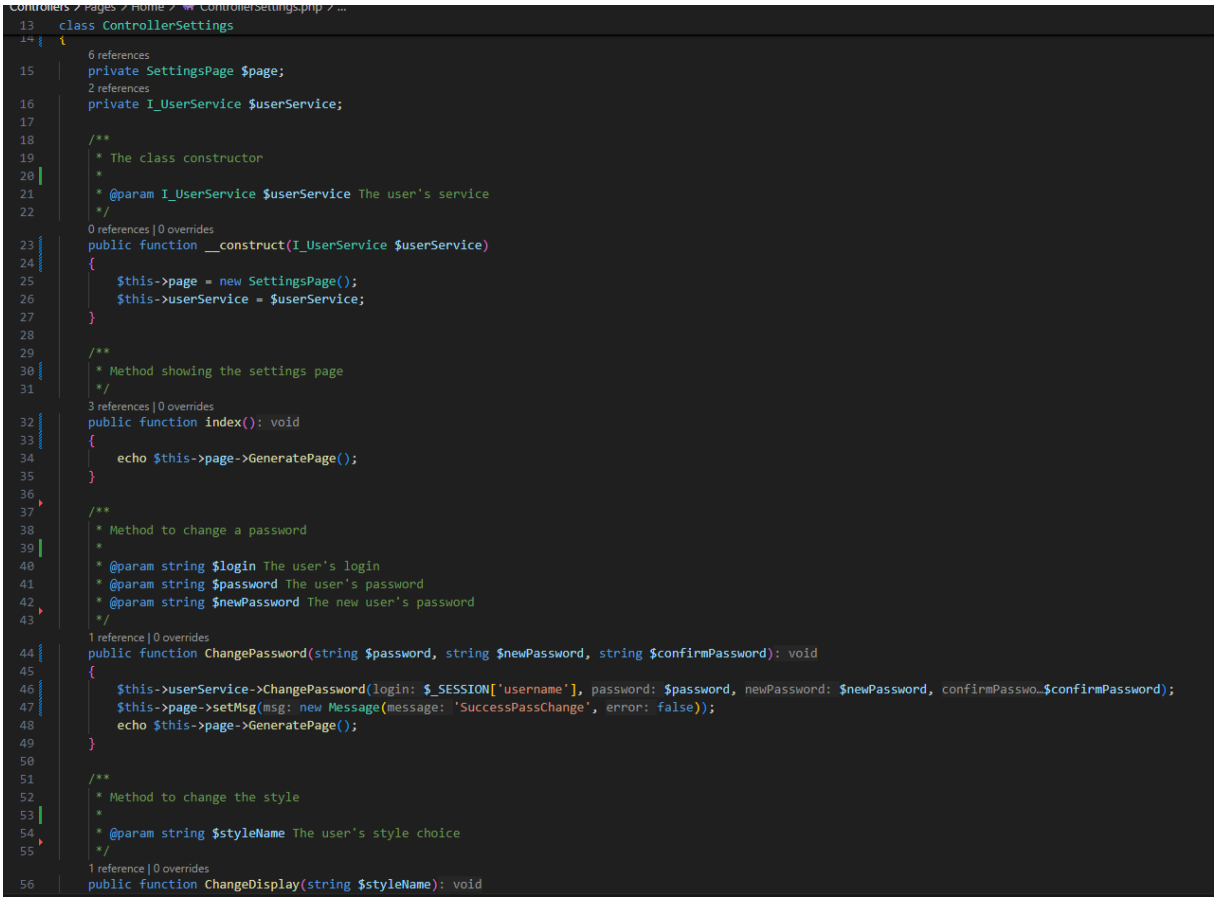


Critères d'audit	Outils utilisés	Descriptions des vérifications effectuées (tests, conditions de test, etc.)	Résultats de l'audit (constats)	Optimisation
C – Format du code source				
C.2 – Aspect du code source				
C.2.1 L'indentation du code est correcte	Visual Studio Code, Webimpress Coding Standard	On effectue une analyse du code sous Webimpress Coding Standard afin de vérifier le nombre d'erreurs d'indentation	NC- On remarque donc la présence de presque 2000 erreurs d'indentation dans ce projet (voir figure ci-dessous)	Après analyse avec Webimpress Coding Standards et différents nettoyages du projet, plus aucune erreur d'indentation n'est finalement détectée.



Critères d'audit	Outils utilisés	Descriptions des vérifications effectuées (tests, conditions de test, etc.)	Résultats de l'audit (constats)	Optimisation
C.2.2 Il n'y a pas d'espace inutiles (blancs) dans le code	Visual Studio Code	Vérification dans chaque fichier qu'aucun grand espace inutile n'est laissé	BP – Aucun réel espace inutile ne se trouve dans le code	/
C.2.3 Le code est facile à lire	Visual Studio Code	On vérifie la présence de commentaires si une fonction n'est pas assez claire sur son propos, la bonne nomenclature de variables et de fonctions, etc.	NC – Les variables sont bien nommées, certains scripts PHP sont correctement commentés (bien que manquant de documentation), mais beaucoup manquent de clarifications et il est difficile de comprendre quelle classe fait quoi (voir figures ci-dessous)	Durant la refonte architecturale du projet, nous avons bien pris le soin, non seulement de rajouter la documentation nécessaire, mais aussi de nommer correctement les différentes variables. La séparation du projet en plusieurs couches rend également plus compréhensible la responsabilité de chacun

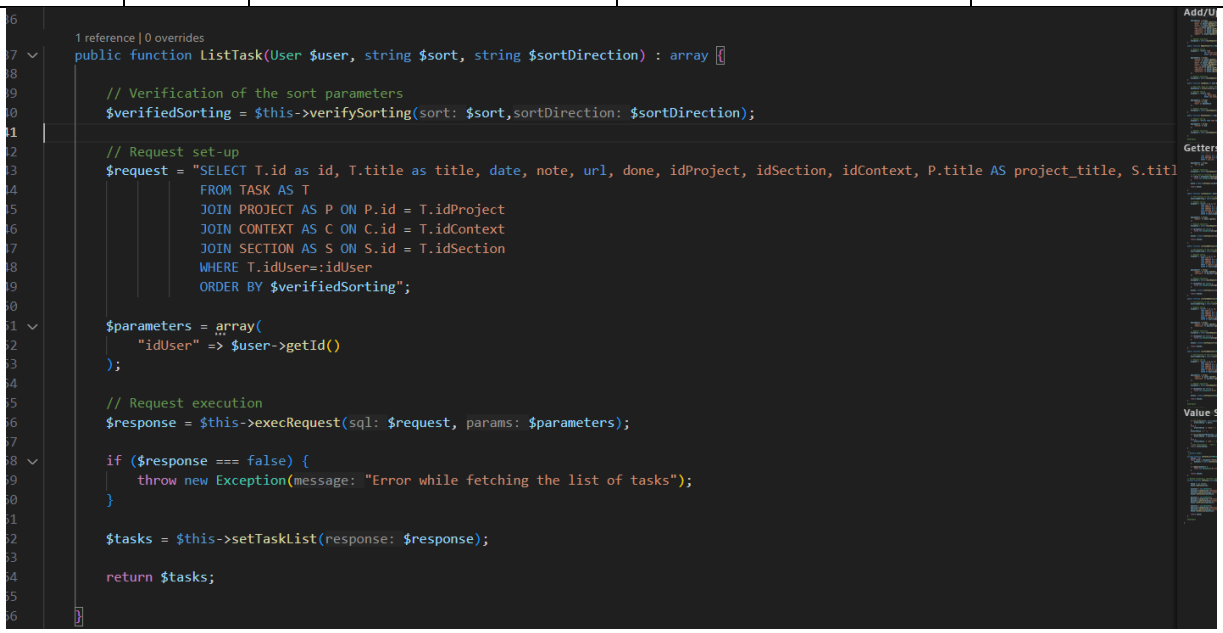



Controllers > Pages > Home > ControllerSettings.php > ...
class ControllerSettings
{
6 references
private SettingsPage $page;
2 references
private I_UserService $userService;
/**
* The class constructor
*
* @param I_UserService $userService The user's service
*/
0 references | 0 overrides
public function __construct(I_UserService $userService)
{
$this->page = new SettingsPage();
$this->userService = $userService;
}
/**
* Method showing the settings page
*/
3 references | 0 overrides
public function index(): void
{
echo $this->page->GeneratePage();
}
/**
* Method to change a password
*
* @param string $login The user's login
* @param string $password The user's password
* @param string $newPassword The new user's password
*/
1 reference | 0 overrides
public function ChangePassword(string $password, string $newPassword, string $confirmPassword): void
{
$this->userService->ChangePassword(login: $_SESSION['username'], password: $password, newPassword: $newPassword, confirmPasswo…$confirmPassword);
$this->page->setMsg(msg: new Message(message: 'SuccessPassChange', error: false));
echo $this->page->GeneratePage();
}
/**
* Method to change the style
*
* @param string $styleName The user's style choice
*/
1 reference | 0 overrides
public function ChangeDisplay(string $styleName): void

Critères d'audit	Outils utilisés	Descriptions des vérifications effectuées (tests, conditions de test, etc.)	Résultats de l'audit (constats)	Optimisation
C.2.4 Le code est dans la langue spécifiée	Visual Studio Code	On vérifie que le code du projet entier est écrit en une même langue	BP – Le code entier est en anglais, jusque dans sa documentation et ses commentaires	Le code est toujours en anglais, y compris dans sa documentation
C.2.5 La taille du code est correcte	Visual Studio Code	Vérification que chaque fonction fait moins de 30 lignes et qu'une classe/un fichier fasse moins de 300 lignes	CF – Chaque fichier fait effectivement moins de 300 lignes mais beaucoup de fonctions dépassent les 30 lignes (voir figure ci-dessous), refactoring et séparation des responsabilités nécessaires	Aux classes de visuel près, chaque classe fait en effet moins de 300 lignes pour la plupart (d'autres pouvant encore être redivisées en plusieurs autres, mais n'ayant pas pu l'être par manque de temps, de même pour différentes fonctions)

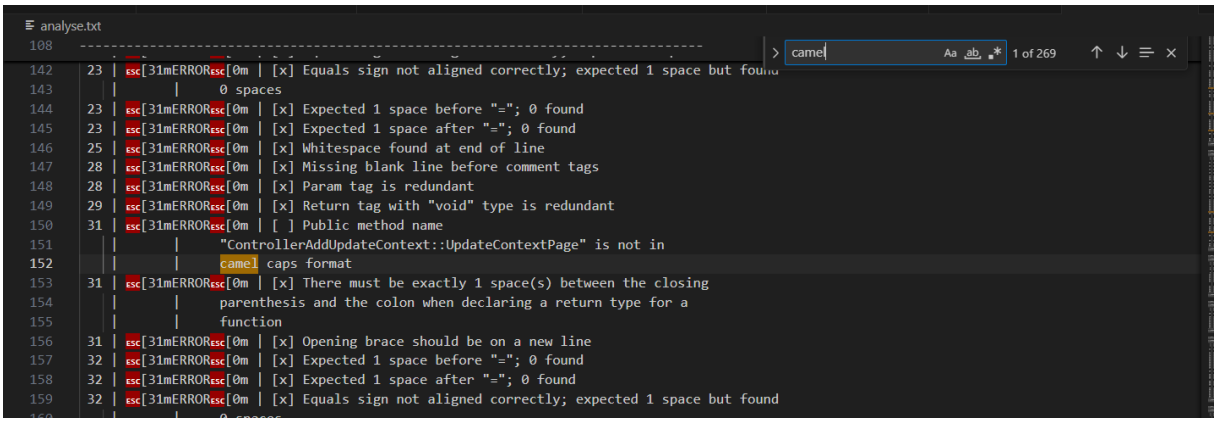
```
11
12  /** * The controller for the website's settings */
    14 references | 0 implementations
13  class ControllerSettings
14  {
        6 references
15      private SettingsPage $page;
        2 references
16      private I_UserService $userService;
17
18      /**…
        0 references | 0 overrides
23      public function __construct(I_UserService $userService)…
27      }
28
29      /**…
        3 references | 0 overrides
32      public function index(): void…
35      }
36
37      /**…
        1 reference | 0 overrides
44      public function ChangePassword(string $password, string $newPassword, string $confirmPassword): void…
49      }
50
51      /**…
        1 reference | 0 overrides
56      public function ChangeDisplay(string $styleName): void…
67      }
68  }
69
```

Critères d'audit	Outils utilisés	Descriptions des vérifications effectuées (tests, conditions de test, etc.)	Résultats de l'audit (constats)	Optimisation
C.2.6 Le code ne présente pas d'élément redondant	Visual Studio Code	Vérification qu'il n'existe pas 2 classes ou fonctions faisant la même chose (principe DRY), vérifier l'usage d'héritage ou d'interface dans ce cas-là	NC – Les récupérations en BDD depuis du code PHP pourraient être factorisées en un même appel sécurisé	Les différentes récupérations en BDD s'opèrent désormais dans une couche DAO, où des requêtes ont pu être factorisées en une, notamment pour ce qui est des tâches

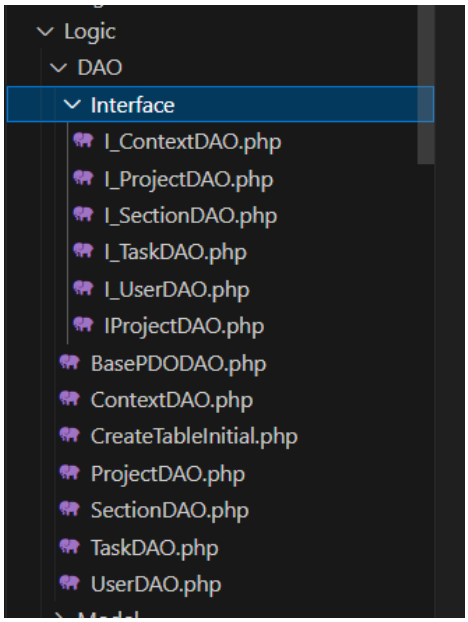


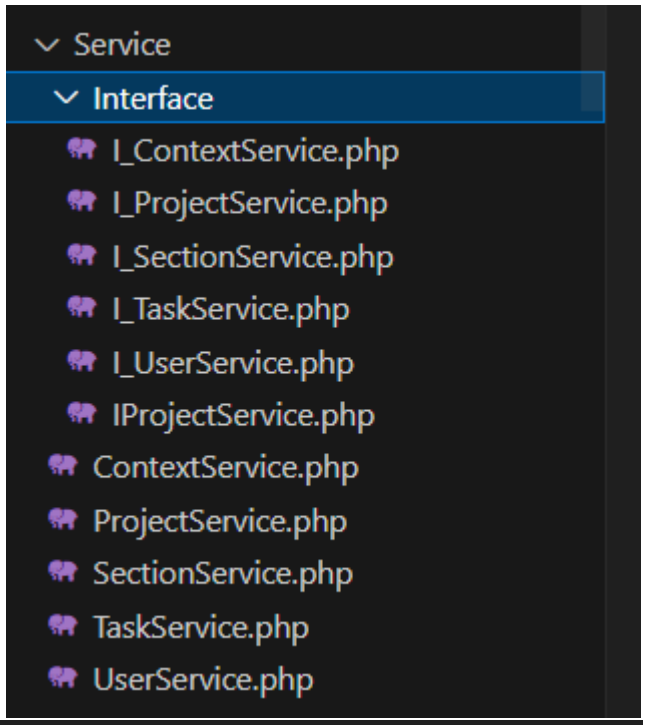
```php
1 reference | 0 overrides
public function ListTask(User $user, string $sort, string $sortDirection) : array {

    // Verification of the sort parameters
    $verifiedSorting = $this->verifySorting(sort: $sort,sortDirection: $sortDirection);

    // Request set-up
    $request = "SELECT T.id as id, T.title as title, date, note, url, done, idProject, idSection, idContext, P.title AS project_title, S.titl
                FROM TASK AS T
                JOIN PROJECT AS P ON P.id = T.idProject
                JOIN CONTEXT AS C ON C.id = T.idContext
                JOIN SECTION AS S ON S.id = T.idSection
                WHERE T.idUser=:idUser
                ORDER BY $verifiedSorting";

    $parameters = array(
        "idUser" => $user->getId()
    );

    // Request execution
    $response = $this->execRequest(sql: $request, params: $parameters);

    if ($response === false) {
        throw new Exception(message: "Error while fetching the list of tasks");
    }

    $tasks = $this->setTaskList(response: $response);

    return $tasks;

}
```

Critères d'audit	Outils utilisés	Descriptions des vérifications effectuées (tests, conditions de test, etc.)	Résultats de l'audit (constats)	Optimisation
C.2.7 La norme de casse du langage utilisé est respectée	Visual Studio Code, Webimpress Coding Standard	Le camel case est respecté (attributs en minuscule, propriété en majuscule au début, pas d'underscores (ils peuvent être remplacé par une majuscule dans le mot), constantes en majuscules)	NC – Par analyse Webimpress Coding Standard, on détecte déjà presque 600 erreurs de camel case dans le code	N'ayant pas nécessairement suivi toutes les bonnes pratiques durant ce projet, il restera des erreurs de camel case, bien qu'elles soient moins nombreuses que dans le projet original



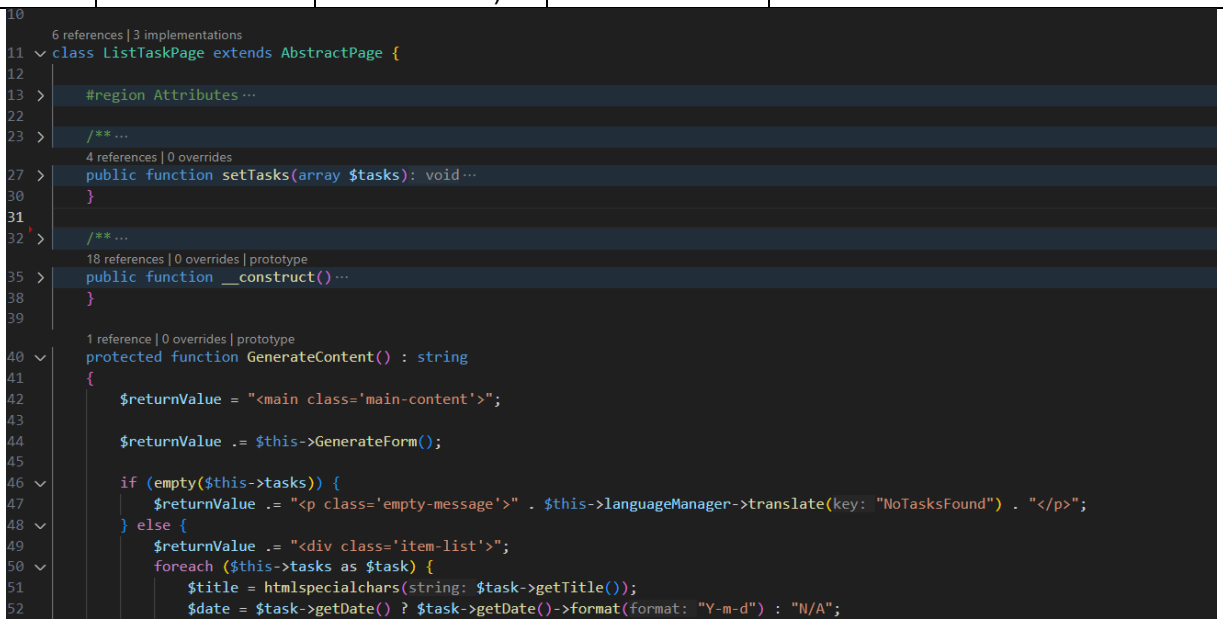
Critères d'audit	Outils utilisés	Descriptions des vérifications effectuées (tests, conditions de test, etc.)	Résultats de l'audit (constats)	Optimisation
D – Principes de qualité logicielle				
D.1 – Principes SOLID				
D.1.1 Le principe de responsabilité unique (S) est respecté	Vscode	On vérifie si chaque fichier fais une seule tache, grâce à ça il serait plus simple de maintenir l'application dans le futur (Une classe, une responsabilité. On cherche différentes classes faisant la même chose ou une faisant plusieurs choses à la fois)	CR - Le fichier display_type.php ne respecte pas le principe S pour la plupart des fichier les appels à la base de données et la gestion de l'affichage sont mélangé dans le code ce qui est contraire au principe S : exemple ci-dessous	Une architecture MVC est en place, veillant à ce que chaque fichier ait une responsabilité unique. L'ensemble est structuré avec des classes PHP, ce qui permet de respecter le principe S De plus, une couche service servira de passerelle entre les couches DAO et Controller, permettant de décharger les responsabilité de l'un ou de l'autre avant d'interagir avec le serveur. (Cf architecture du projet pour plus de détails)
D.1.2 Le principe ouvert-fermé (O) est respecté	Vs code	On vérifie s'il y a une isolation des couches et si pour ajouter une fonctionnalité on peut	CR - il n'y a aucune interface ni de séparation des package di on veut ajouter une fonctionnalité on est obligé de modifier du code de partout pour que ça marche	La partie métier est isolée à l'aide d'interfaces, ce qui permet d'ajouter de nouvelles fonctionnalités facilement, sans impacter le reste du système. Ceci permet de respecter le principe O image d'illustration Une bibliothèque d'injection de dépendances (PHP-DI) est également installée afin de pouvoir facilement remplacer un service ou un DAO à l'avenir



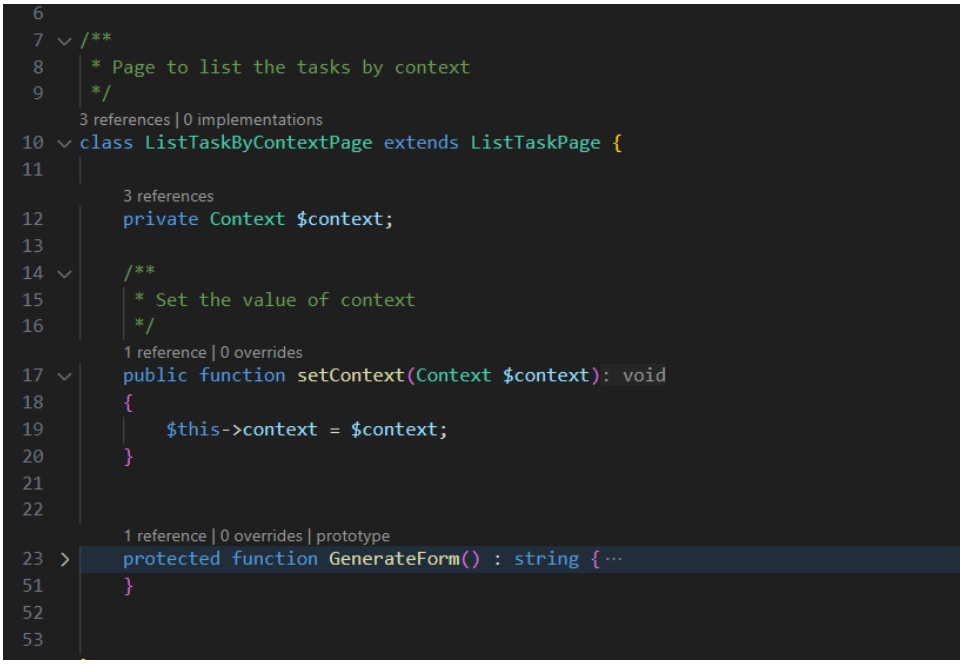



```php

// Configuration of the dependecies injection
return [

    // DAO
    I_TaskDAO::class => autowire(className: TaskDAO::class),
    I_ProjectDAO::class => autowire(className: ProjectDAO::class),
    I_SectionDAO::class => autowire(className: SectionDAO::class),
    I_ContextDAO::class => autowire(className: ContextDAO::class),
    I_UserDAO::class => autowire(className: UserDAO::class),

    // Services
    I_TaskService::class => autowire(className: TaskService::class),
    I_ContextService::class => autowire(className: ContextService::class),
    I_SectionService::class => autowire(className: SectionService::class),
    I_ProjectService::class => autowire(className: ProjectService::class),
    I_UserService::class => autowire(className: UserService::class)

];


?>
```

D.1.3 Le principe de substitution de Liskov (L) est respecté	Visual Studio Code	Vérification qu'en cas d'héritage, le module de bas peut remplacer celui de haut niveau sans en changer le fonctionnement (exemple des exceptions levées par les classes filles)	CR – Pas d'objets donc à mettre en place dans le projet	La transformation du projet pour utiliser des classes PHP aurait pu poser un problème vis-à-vis du principe L. Cependant, ce principe SOLID a bien été pris en compte afin d'éviter toute erreur à ce niveau. On retrouve particulièrement des notions d'héritage du côté des vues (ListTaskBySection héritant de ListTask par exemple) et l'un peut aisément remplacer l'autre

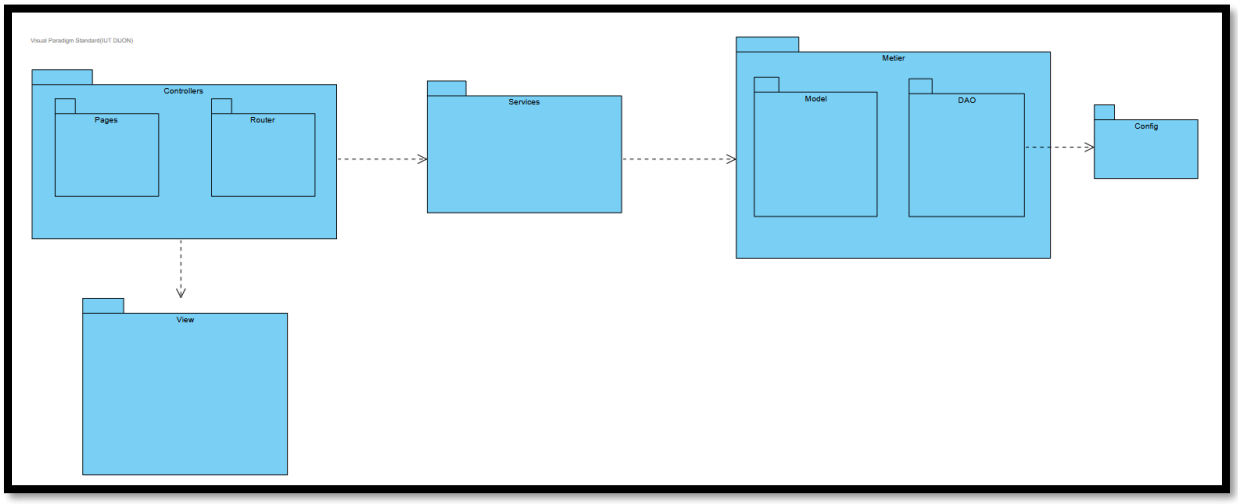


```
6 references | 3 implementations
class ListTaskPage extends AbstractPage {

    #region Attributes ···

    /** ···
    4 references | 0 overrides
    public function setTasks(array $tasks): void ···
    }

    /** ···
    18 references | 0 overrides | prototype
    public function __construct() ···
    }

    1 reference | 0 overrides | prototype
    protected function GenerateContent() : string
    {
        $returnValue = "<main class='main-content'>";

        $returnValue .= $this->GenerateForm();

        if (empty($this->tasks)) {
            $returnValue .= "<p class='empty-message'>" . $this->languageManager->translate(key: "NoTasksFound") . "</p>";
        } else {
            $returnValue .= "<div class='item-list'>";
            foreach ($this->tasks as $task) {
                $title = htmlspecialchars(string: $task->getTitle());
                $date = $task->getDate() ? $task->getDate()->format(format: "Y-m-d") : "N/A";
```

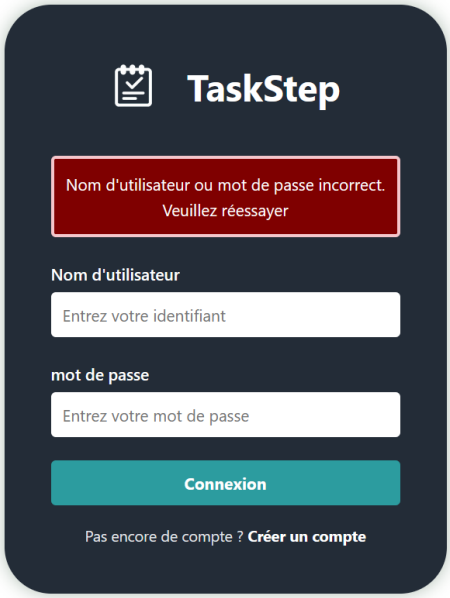


```
/**
 * Page to list the tasks by context
 */
3 references | 0 implementations
class ListTaskByContextPage extends ListTaskPage {

    3 references
    private Context $context;

    /**
     * Set the value of context
     */
    1 reference | 0 overrides
    public function setContext(Context $context): void
    {
        $this->context = $context;
    }


    1 reference | 0 overrides | prototype
    protected function GenerateForm() : string {···
    }
```

D.1.4 Le principe de ségrégation des interfaces (I) est respecté	Visual Studio Code	Une classe n'implémente pas une interface dont elle n'a pas besoin de toutes les méthodes, éviter les interfaces pour tout, qui ont trop de méthodes (principe S)	CR – Pas d'objets donc à mettre en place dans le projet	L'ajout d'interface aurait pu poser souci quant au respect du principe I mais nous l'avons bien respecté, comme démontré sur le point D.1.2. sur le principe O
D.1.5 Le principe d'inversion des dépendances (D) est respecté		(Code de haut niveau ne dépend pas de bas niveau (métier qui parle à IHM directement))	CR – Pas d'objets donc à mettre en place dans le projet	Le principe D garantit que les dépendances entre classes vont dans le bon sens et qu'il n'existe pas de dépendances circulaires. Ce point a également été pris en compte dans notre conception (cf. diagramme des classes), ayant une claire séparation des couches : Controllers a accès aux Services qui ont accès aux DAO et aux Models, Controllers ont également accès aux Vues qui ont accès aux Models. Cette séparation des couches évite donc des dépendances circulaires



Critères d'audit	Outils utilisés	Descriptions des vérifications effectuées (tests, conditions de test, etc.)	Résultats de l'audit (constats)	Optimisation
D – Principes de qualité logicielle				
D.2 – Gestion des erreurs				
D.2.1 – Les cas d'erreurs sont traités à l'aide des exceptions	VsCode	Test via l'IHM et analyse du code pour trouver des erreurs gérées	CR – Aucune présence d'exception	Les exceptions sont directement capturées dans le router de sorte que l'on n'ait aucune exception n'étant pas gérée dans l'application
D.2.2 Le point d'entrée du programme capture les exceptions non encore capturées	VsCode	Test via l'IHM et analyse du code pour trouver des erreurs gérées et trouver des blocs de try catch	CR – Vu qu'il n'y a aucune exception les seules erreurs affichées viennent d'if/else	
D.2.3 Les exceptions ne sont pas muselées mais capturées pour être traitées (catchs vides, à documenter si c'est volontaire)	VsCode	Test via l'IHM et analyse du code	CR - Aucune présences de try / cacth sauf dans la bibliothèque	



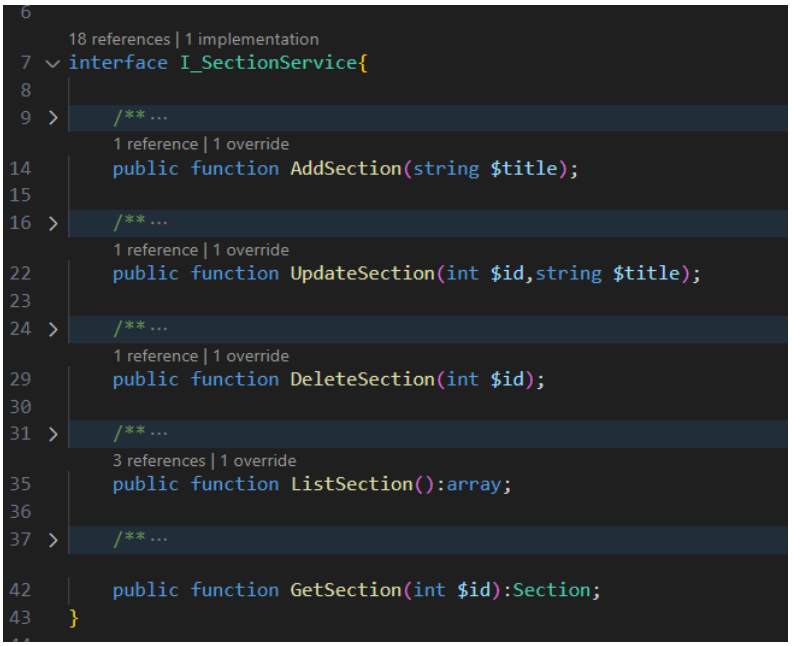
Critères d'audit	Outils utilisés	Descriptions des vérifications effectuées (tests, conditions de test, etc.)	Résultats de l'audit (constats)	Optimisation
D – Principes de qualité logicielle				
D.3 – Règles de codage				
D.3.1 – Il n'y a pas de « valeur magique » dans le code	Visual Studio Code	Il n'y a pas de valeurs arbitraires sorties de nulle part dans le code (limites de boucles par exemple)	CF – Une valeur magique se trouve au milieu des fonctions (display_frontpage()) et devrait donc être échappée en dehors de la fonction (variable globale par exemple) ou en paramètre optionnel avec valeur par défaut (voir figures ci-dessous)	Cette valeur magique n'existe maintenant plus et les variables globales de l'application sont contenues dans un fichier séparé pour l'application

Defines.php > ...

```php
<?php

#region Connexion

// Time to wait before trying another connexion again
define(constant_name: "TENTATIVES_TIMEOUT",value: 600);

// Number of tries before waiting the time above
define(constant_name: "TENTATIVES_CONNEXIONS",value: 5);

// The time you can remain connected
define(constant_name: "INACTIVITY_TIMEOUT",value: 1200);

#endregion
```

Critères d'audit	Outils utilisés	Descriptions des vérifications effectuées (tests, conditions de test, etc.)	Résultats de l'audit (constats)	Optimisation
D.3.2 – chaque fonction ne possède qu'un return, sur la dernière ligne de la fonction	Visual Studio Code	Vérification qu'aucune fonction ne possède plus d'un return, ce dernier devant être placé en dernière ligne de fonction	BP – Aucune fonction ne possède plus d'un return dans le projet	De même, après réarrangement du projet, chaque fonction ne possède qu'un return
D.3.3 – aucune procédure ne contient de return	Visual Studio Code	Vérification qu'il n'y a pas de returns en procédure	BP – Il n'y a pas de returns en procédures du projet	Les fonctions ne renvoyant rien ne possèdent pas de return
D.3.4 – il n'y a aucun « goto » ou autre rupture du flux d'exécution	Visual Studio Code	Vérification qu'aucune rupture du flux n'a lieu dans le projet, qu'il n'y a pas de goto renvoyant à une autre ligne de code en particulier, ni de break en milieu de fonction	BP – Il n'y a pas de rupture du flux	Aucune rupture de flux n'a été implémentée du projet

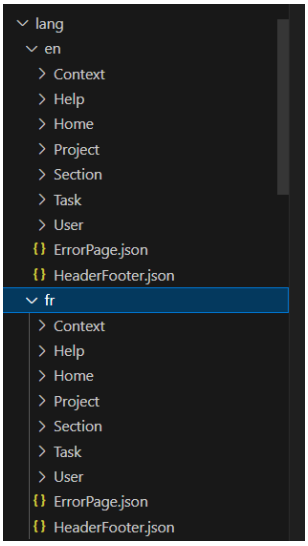
D.3.5 – Le nom des variables est pertinent	Visual Studio Code	On vérifie que les noms donnés sont pertinents comme les noms de variable ou autre	NC – Certains noms ne sont absolument pas pertinents et sont non professionnels parfois par manque d'inspiration le nom est le même avec un numéro ce qui rend le code encore plu illisible	Chaque variable a été nommée de façon cohérente et logique avec l'application
D.3.6 – le nom des opérations (fonctions, procédures) est pertinent	Visual studio code	Vérification que les noms des fonctions est pertinent et clair	NC – Certains noms de fonction ne sont pas très clair et ne possède pas de nomenclature	De même, chaque fonction et procédure est nommée de façon pertinente, les rendant reconnaissables et utilisables

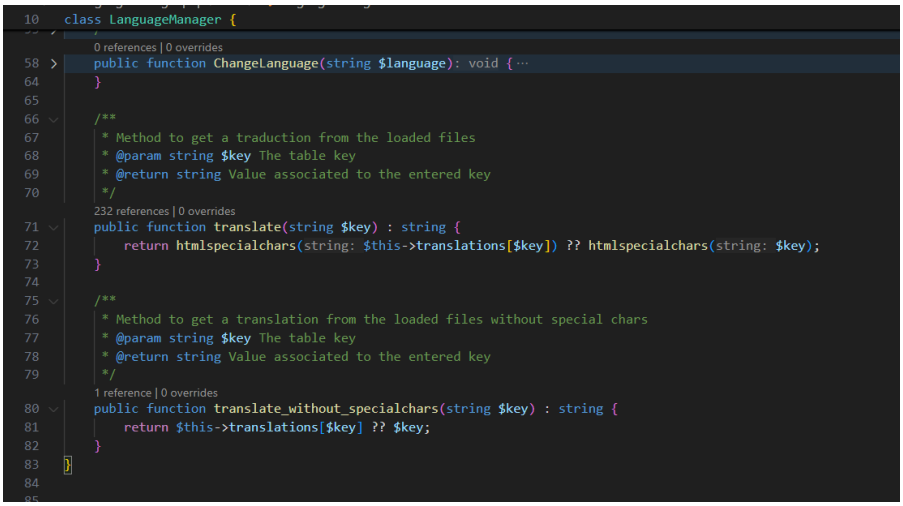


```php
2 references | 0 implementations
class ProjectCreateRoute extends Route {
    3 references
    private ControllerAddUpdateProject $controllerProject;
    /**
     * constructor
     * @param \Controllers\Pages\Project\ControllerAddUpdateProject $controllerProject project controll
     */
    1 reference | 0 overrides
    public function __construct(ControllerAddUpdateProject $controllerProject){
        $this->controllerProject = $controllerProject;
    }
    #region methodes
    1 reference | 0 overrides | prototype
    public function get(array $params=[]): void{
        $this->controllerProject->AddProjectPage();
    }

    1 reference | 0 overrides | prototype
    public function post(array $params=[]): void{
        $title = parent::getParam(array: $params,paramName: "title",canBeEmpty: false);
        $this->controllerProject->AddProject(title: $title);
    }
    #endregion
```

```php
/**
 * Abstract class for a route
 */
131 references | 31 implementations
abstract class Route {

    /**
     * Method to act on a road, calling the method GET or POST based on the given parameter
     * @param array $params Parameters
     * @param string $method Action method (get by default)
     * @return void
     */
    2 references | 0 overrides
    public function action(array $params = [], string $method='GET'): void {

        try {
            switch ($method) {
                case 'GET':
                    $this->get(params: $params);
                    break;

                case 'POST':
                    $this->post(params: $params);
                    break;
            }
        }
        catch (Exception $e) {
            $controller = new ControllerError();
            $controller->index(message: $e->getMessage());
        }


    }
```

Critères d'audit	Outils utilisés	Descriptions des vérifications effectuées (tests, conditions de test, etc.)	Résultats de l'audit (constats)	Temps estimé d'optimisation
D.3.7 – il n'y a pas de code « mort » (commenté, non atteignable…) (fonctions jamais appelées, ou alors code qui n'est jamais exécuté du programme)	Visual Studio Code	Vérification qu'il n'y a pas de code commenté ou non atteignable, de fonctions jamais appelées ou de code qui n'est jamais exécuté du programme	CF – Plusieurs zones de code sont commentées	Aucun code mort n'a été laissé dans ce nouveau projet
E – Pile technique				
E.1 – Langage utilisé				
E.1.1 – Le(s) langage(s) utilisés sont pertinents pour le problème donné (Exemple non pertinents : application en Python, IA en C#)	Visual Studio Code, XAMPP	Vérifications que les langages utilisés ne sont pas hors-sujet et sont pertinents à utiliser.	CF – Comme il s'agit d'un site web, le code PHP est adapté dans l'ensemble à de l'interaction côté serveur et à la communication avec une Base de Données Pour la base de données la communication se fait par l'utilisation de MySQL Pour la traduction, quant à elle, elle n'est pas réalisée dans le bon langage, elle est réalisée en PHP alors qu'elle devrait être en .json	Le reste des éléments étant corrects dans l'ensemble, nous avons simplement changé le système de traduction afin qu'il utilise des fichiers .json pour traduire le site en différentes langues





E.1.2 – La version utilisée du langage est supportée et maintenue (Ex : page en PHP5, failles non corrigées aujourd'hui)	Visual Studio Code, README	Vérification que le langage de programmation est supporté par les navigateurs web et maintenu encore aujourd'hui (lecture du README pour trouver les versions)	CR – La version utilisée est php 7.0 qui n'est plus maintenue, il faut mettre à jour les différentes méthodes n'étant plus d'actualité (PDO pour interactions BDD, POO, ...)	Le code a pu être mis à jour et codé en PHP 8.2, permettant ainsi des interactions plus sécurisées avec la base de données, évitant notamment les injections SQL
E.2 – Outils tiers				
E.2.1 – Les bibliothèques/frameworks utilisés sont pertinents et réellement utiles	Visual Studio Code	Analyse du code, on vérifie que les bibliothèques utilisées servent réellement et n'ont pas un trop grand nombre de fonctionnalité pour un usage trop petit	NC – L'unique bibliothèques utilisée était utile en 2006 mais n'est plus utiles actuellement est pris en charge nativement par HTML depuis 2015 (voir figure ci-dessous)	La bibliothèque des dates en HTML a donc été retirée au profit d'une simple balise <input type="date"> Une seule bibliothèque sera désormais utilisée, servant à l'injection de dépendances en PHP (PHP-DI 7.0)
E.2.2 – Les bibliothèques/frameworks utilisés sont dans une version supportée et maintenue	Visual Studio Code	Vérification que la ou les bibliothèques du projet sont encore d'actualité et maintenue(s), si besoin, les mettre à jour	NC – La seule bibliothèque utilisée n'est plus maintenue.	PHP-DI est encore actuellement maintenu et supporté, fonctionnant parfaitement bien avec PHP 8.2

IV. Tableau des tâches d'optimisation réalisées

Le tableau suivant liste les tâches d'optimisation qui étaient définies par l'audit de qualité logicielle, la dernière colonne indiquera en vert si la tâche a pu être réalisée, en orange si elle ne l'est que partiellement, en rouge si elle n'a pas pu l'être

Recommandations	Priorité	Complexité	Effort	gravité	Fait / Non fait
tutoriel hébegement local	1	1	1	1	
changement de langue	2	3	2	2	
tests unitaires	2	3	2	1	
présence de valeur arbitraire	2	2	2	2	
code mort	2	1	1	1	
bibliothèque pour la date n'est plus utile maintenant	2	1	1	1	
bibliothèque non maintenue	2	1	1	1	
mauvaise gestion des cas particulier	3	3	2	2	
erreur et warning dans le code	3	2	3	2	
non typage	3	1	2	2	
indentation	3	1	1	1	
code difficile à lire	3	3	2	2	
non respect du camel case	3	2	2	1	
traduction en PHP au lieu de json	3	3	2	2	
documentation	4	1	2	1	
fonction trop longue	4	2	2	2	
non présence d'exception	4	3	2	3	
non présence de try catch (sauf dans la bibliothèque)	4	2	3	3	
noms impertinents (variables et fonctions)	4	4	3	3	
mettre à jour la version de PHP	4	4	3	3	
diagramme cas d'usage	5	3	2	2	
mélange requête SQL et code PHP	5	3	2	3	
architecture	6	3	3	3	
principe SOLID non respecté	6	6	4	3	

Par manque de temps, nous n'avons pas pu analyser chaque cas particulier du projet et n'avons pas pu résoudre tous les warnings du projet, mais le reste des éléments, notamment tous ceux à priorité très élevée, a pu être corrigé, rendant donc globalement l'application beaucoup plus qualitative en termes de code.

V. Synthèse d'optimisation

L'application a pu être jugée selon les principes SOLID, les bonnes pratiques de programmation et le PSR (PHP Standards Recommandations) durant l'audit de qualité logicielle. En découlait donc les soucis d'architecture, de documentations, de tests ou même de respect des principes évoqués précédemment.

Nous avons donc pris la décision de refaire intégralement l'architecture de l'application étant donné que Taskstep n'avait originellement aucune structure, ni même de programmation orientée objet. Ceci se faisant dans le respect des principes SOLID, en documentant chaque classe, interface et fonction publique, et en respectant chacun des critères décrits durant tout cet audit d'optimisation.

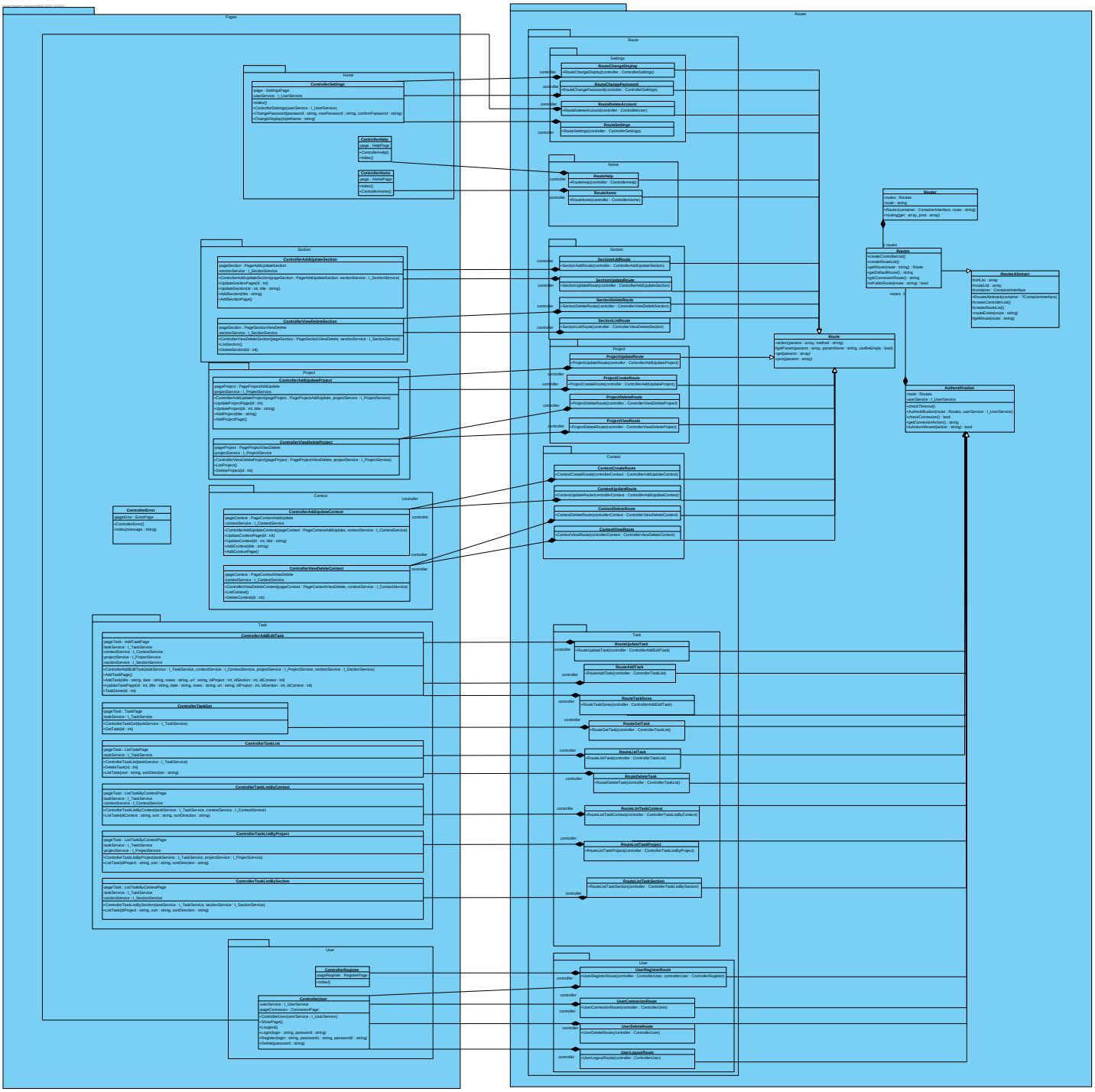
Pour ce qui est de la sécurité et de la gestion des données dans le code, il aura été réalisé de sorte qu'aucune injection SQL ne soit possible. Des tests unitaires de la couche DAO, donc pour les interactions avec la base de données, auront également été réalisés afin de pouvoir tester la validité des tables du projet.

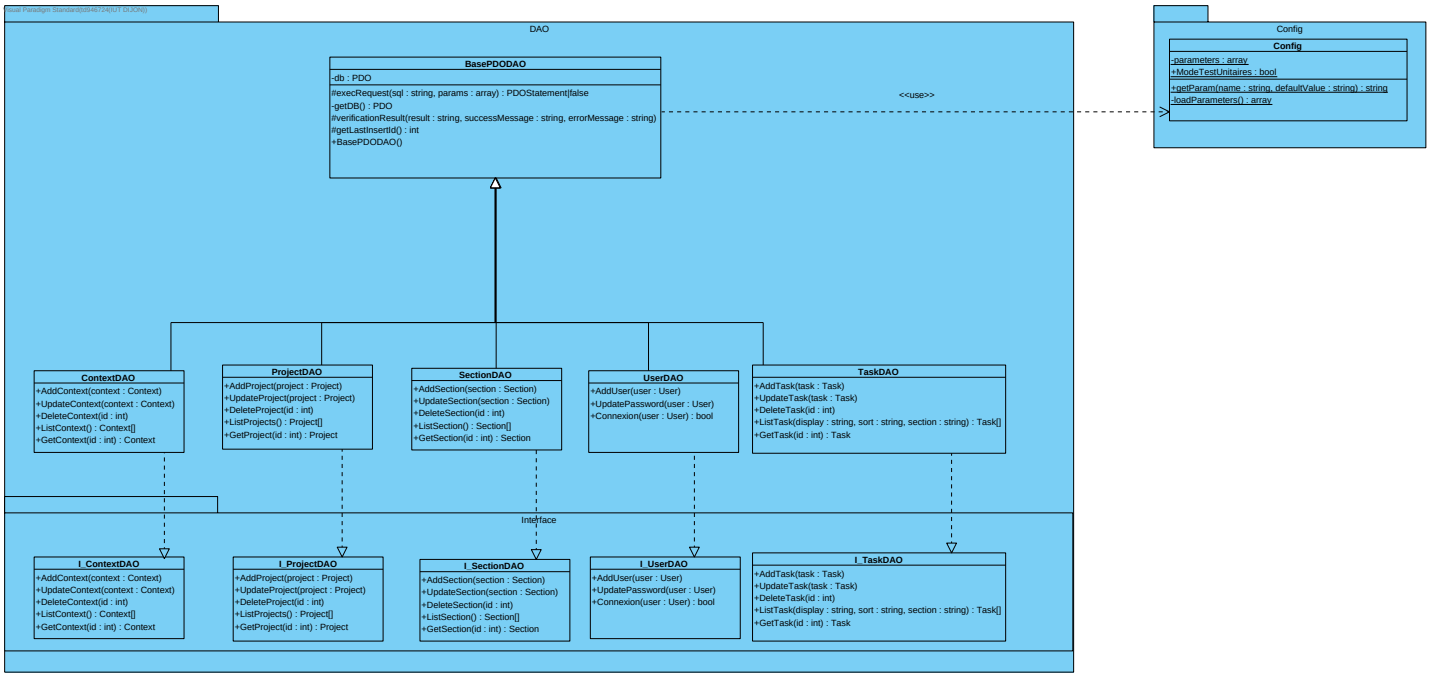
La traduction du site est désormais gérée par différents fichiers JSON, rendant la mise en place d'une nouvelle langue beaucoup plus facile.

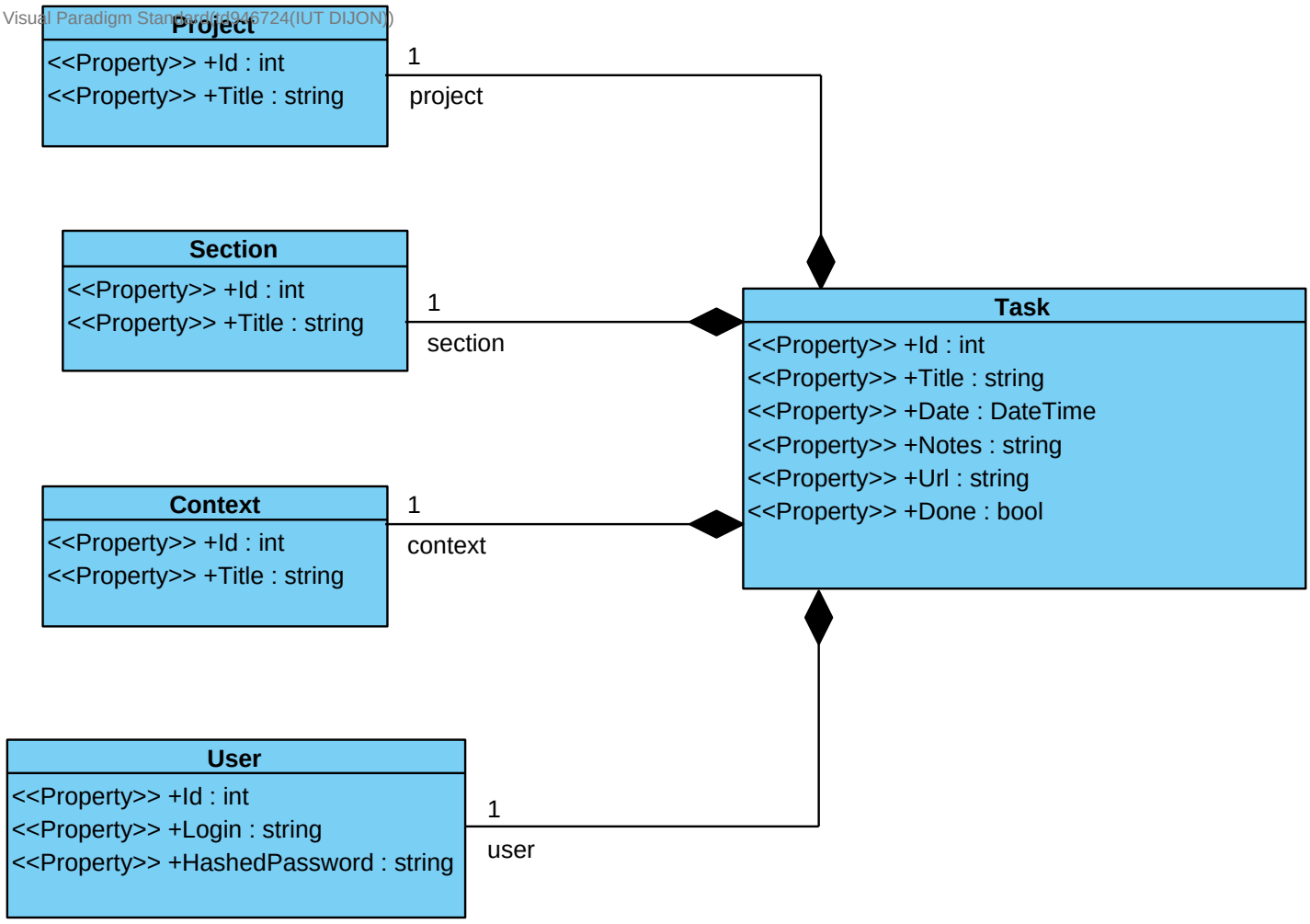
Globalement, ces changements permettent de rendre l'application beaucoup plus modulaire et évolutive qu'elle ne l'était auparavant, le respect des bonnes pratiques de programmation en étant en grande partie responsable. D'autres optimisations secondaires n'ont pas pu être finalisées dans les temps (comme la gestion de cas particuliers ou l'implémentation de tests unitaires hors-couche DAO) et pourraient être réalisés lors d'une poursuite du projet.

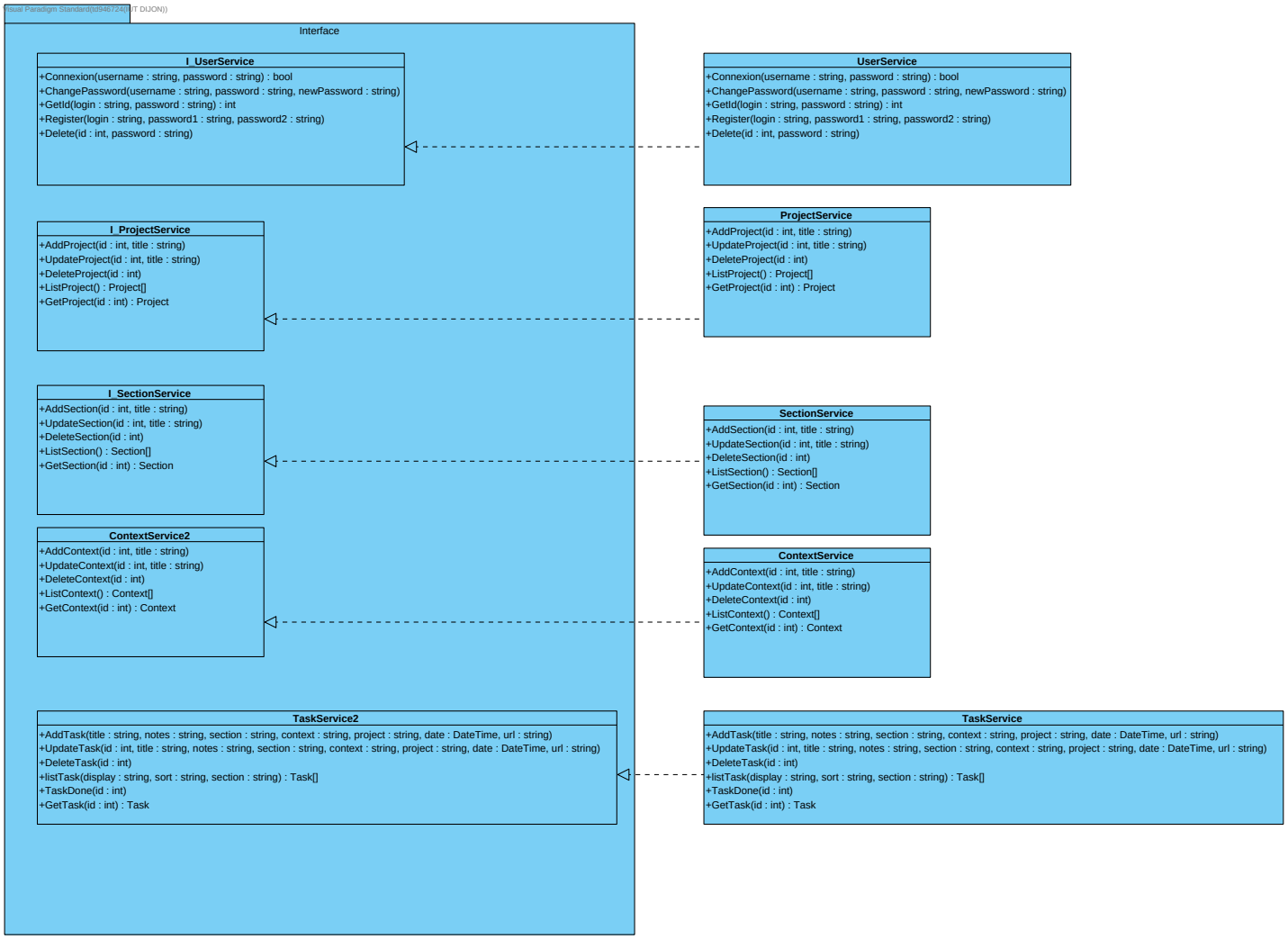
VI. Annexe : UML du projet

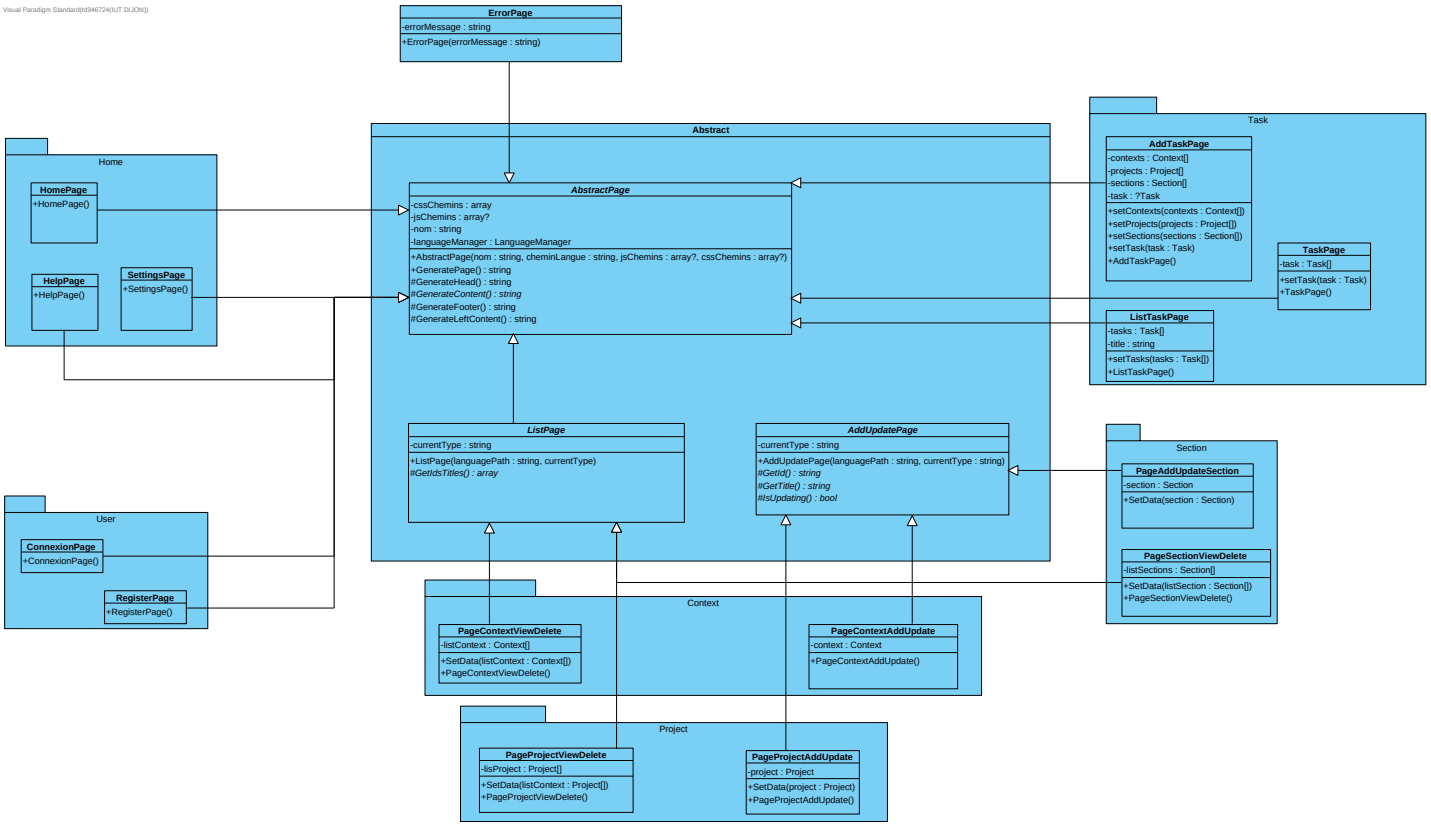
Ci-dessous se trouvera le nouvel UML du projet dans son intégralité

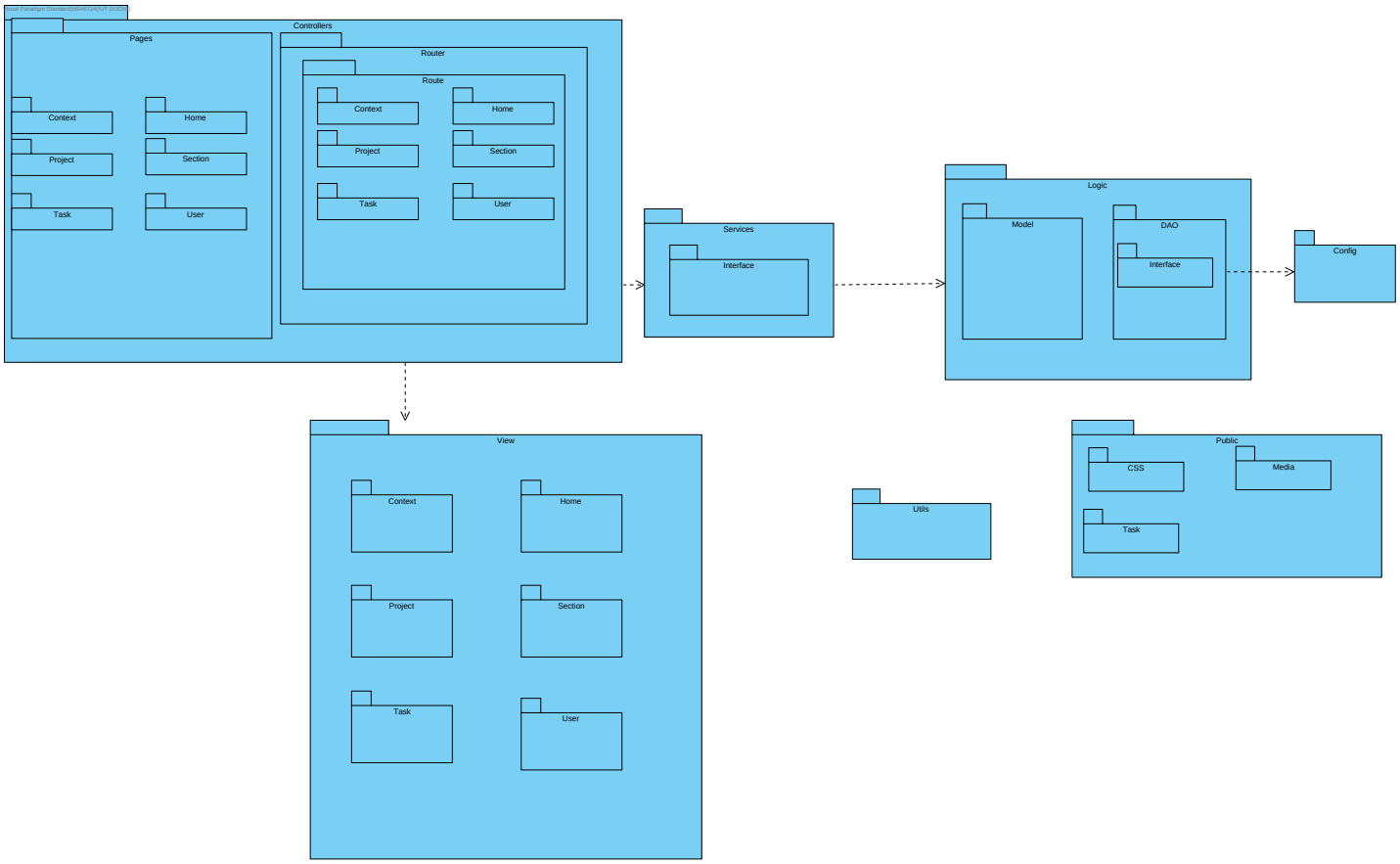

Pages
Home
ControllerSettings
page : SettingsPage
userService : I_UserService
+Index()
+ControllerSettings(userService : I_UserService)
+ChangePassword(password : string, newPassword : string, confirmPassword : string)
+ChangeDisplay(styleName : string)
ControllerHelp
page : HelpPage
+ControllerHelp()
+Index()
ControllerHome
page : HomePage
+Index()
+ControllerHome()
Section
ControllerAddUpdateSection
pageSection : PageAddUpdateSection
sectionService : I_SectionService
+ControllerAddUpdateSection(pageSection : PageAddUpdateSection, sectionService : I_SectionService)
+UpdateSectionPage(id : int)
+UpdateSection(id : int, title : string)
+AddSection(title : string)
+AddSectionPage()
ControllerViewDeleteSection
pageSection : PageSectionViewDelete
sectionService : I_SectionService
+ControllerViewDeleteSection(pageSection : PageSectionViewDelete, sectionService : I_SectionService)
+ListSection()
+DeleteSection(id : int)
Project
ControllerAddUpdateProject
pageProject : PageProjectAddUpdate
projectService : I_ProjectService
+ControllerAddUpdateProject(pageProject : PageProjectAddUpdate, projectService : I_ProjectService)
+UpdateProjectPage(id : int)
+UpdateProject(id : int, title : string)
+AddProject(title : string)
+AddProjectPage()
ControllerViewDeleteProject
pageProject : PageProjectViewDelete
projectService : I_ProjectService
+ControllerViewDeleteProject(pageProject : PageProjectViewDelete, projectService : I_ProjectService)
+ListProject()
+DeleteProject(id : int)
Context
controller
ControllerAddUpdateContext
pageContext : PageContextAddUpdate
contextService : I_ContextService
+ControllerAddUpdateContext(pageContext : PageContextAddUpdate, contextService : I_ContextService)
+UpdateContextPage(id : int)
+UpdateContext(id : int, title : string)
+AddContext(title : string)
+AddContextPage()
ControllerViewDeleteContext
pageContext : PageContextViewDelete
contextService : I_ContextService
+ControllerViewDeleteContext(pageContext : PageContextViewDelete, contextService : I_ContextService)
+ListContext()
+DeleteContext(id : int)
ControllerError
pageError : ErrorPage
+ControllerError()
+index(message : string)
Task
ControllerAddEditTask
pageTask : AddTaskPage
taskService : I_TaskService
contextService : I_ContextService
projectService : I_ProjectService
sectionService : I_SectionService
+ControllerAddEditTask(taskService : I_TaskService, contextService : I_ContextService, projectService : I_ProjectService, sectionService : I_SectionService)
+AddTaskPage()
+AddTask(title : string, date : string, notes : string, url : string, idProject : int, idSection : int, idContext : int)
+UpdateTaskPage(id : int, title : string, date : string, notes : string, url : string, idProject : int, idSection : int, idContext : int)
+TaskDone(id : int)
ControllerTaskGet
pageTask : TaskPage
taskService : I_TaskService
+ControllerTaskGet(taskService : I_TaskService)
+GetTask(id : int)
ControllerTaskList
pageTask : ListTaskPage
taskService : I_TaskService
+ControllerTaskList(taskService : I_TaskService)
+DeleteTask(id : int)
+ListTask(sort : string, sortDirection : string)
ControllerTaskListByContext
pageTask : ListTaskByContextPage
taskService : I_TaskService
contextService : I_ContextService
+ControllerTaskListByContext(taskService : I_TaskService, contextService : I_ContextService)
+ListTaskByContext : string, sort : string, sortDirection : string)
ControllerTaskListByProject
pageTask : ListTaskByContextPage
taskService : I_TaskService
projectService : I_ProjectService
+ControllerTaskListByProject(taskService : I_TaskService, projectService : I_ProjectService)
+ListTaskByProject : string, sort : string, sortDirection : string)
ControllerTaskListBySection
pageTask : ListTaskByContextPage
taskService : I_TaskService
sectionService : I_SectionService
+ControllerTaskListBySection(taskService : I_TaskService, sectionService : I_SectionService)
+ListTaskByProject : string, sort : string, sortDirection : string)
User
ControllerRegister
pageRegister : RegisterPage
+index()
ControllerUser
-userService : I_UserService
-pageConnexion : ConnexionPage
+ControllerUser(userService : I_UserService)
+ShowPage()
+Logout()
+Login(login : string, password : string)
+Register(login : string, password1 : string, password2 : string)
+Delete(password : string)
Router
Route
Settings
RouteChangeDisplay
+RouteChangeDisplay(controller : ControllerSettings)
RouteChangePassword
+RouteChangePassword(controller : ControllerSettings)
RouteDeleteAccount
+RouteDeleteAccount(controller : ControllerUser)
RouteSettings
+RouteSettings(controller : ControllerSettings)
Home
RouteHelp
+RouteHelp(controller : ControllerHelp)
RouteHome
+RouteHome(controller : ControllerHome)
Section
SectionAddRoute
+SectionAddRoute(controller : ControllerAddUpdateSection)
SectionUpdateRoute
+SectionUpdateRoute(controller : ControllerAddUpdateSection)
SectionDeleteRoute
+SectionDeleteRoute(controller : ControllerViewDeleteSection)
SectionListRoute
+SectionListRoute(controller : ControllerViewDeleteSection)
Project
ProjectUpdateRoute
+ProjectUpdateRoute(controller : ControllerAddUpdateProject)
ProjectCreateRoute
+ProjectCreateRoute(controller : ControllerAddUpdateProject)
ProjectDeleteRoute
+ProjectDeleteRoute(controller : ControllerViewDeleteProject)
ProjectViewRoute
+ProjectDeleteRoute(controller : ControllerViewDeleteProject)
Context
ContextCreateRoute
+ContextCreateRoute(controllerContext : ControllerAddUpdateContext)
ContextUpdateRoute
+ContextUpdateRoute(controllerContext : ControllerAddUpdateContext)
ContextDeleteRoute
+ContextDeleteRoute(controllerContext : ControllerViewDeleteContext)
ContextViewRoute
+ContextViewRoute(controllerContext : ControllerViewDeleteContext)
Task
RouteUpdateTask
+RouteUpdateTask(controller : ControllerAddEditTask)
RouteAddTask
+RouteAddTask(controller : ControllerTaskList)
RouteTaskDone
+RouteTaskDone(controller : ControllerAddEditTask)
RouteGetTask
+RouteGetTask(controller : ControllerTaskList)
RouteListTask
+RouteListTask(controller : ControllerTaskList)
RouteDeleteTask
+RouteDeleteTask(controller : ControllerTaskList)
RouteListTaskContext
+RouteListTaskContext(controller : ControllerTaskListByContext)
RouteListTaskProject
+RouteListTaskProject(controller : ControllerTaskListByProject)
RouteListTaskSection
+RouteListTaskSection(controller : ControllerTaskListBySection)
User
UserRegisterRoute
+UserRegisterRoute(controller : ControllerUser, controllerUser : ControllerRegister)
UserConnexionRoute
+UserConnexionRoute(controller : ControllerUser)
UserDeleteRoute
+UserDeleteRoute(controller : ControllerUser)
UserLogoutRoute
+UserLogoutRoute(controller : ControllerUser)
Router
-routes : Routes
-route : string
+Router(container : ContainerInterface, route : string)
+routing(get : array, post : array)
routes
Routes
+createControllerList()
+createRoutesList()
+getRoute(route : string) : Route
+getDefaultRoute() : string
+getConnexionRoute() : string
+isPublicRoute(route : string) : bool
RoutesAbstract
#ctrlList : array
#routesList : array
#container : ContainerInterface
+RoutesAbstract(container : ?ContainerInterface)
#createControllerList()
#createRoutesList()
+routeExists(route : string)
#getRoute(route : string)
Route
+action(params : array, method : string)
#getParams(params : array, paramName : string, canBeEmpty : bool)
+get(params : array)
+post(params : array)
routes
1
Authentification
-route : Routes
-userService : I_UserService
+checkTimeout()
+isAuthentificationRoute(route : Routes, userService : I_UserService)
+checkConnexion() : bool
-getConnexionAction() : string
+isActionAllowed(action : string) : bool


Visual Paradigm Standard(ddd46724@UT DI JON))
DAO
BasePDODAO
-db : PDO
#execRequest(sql : string, params : array) : PDOStatement|false
-getDB() : PDO
#verificationResult(result : string, successMessage : string, errorMessage : string)
#getLastInsertId() : int
+BasePDODAO()
<<use>>
Config
Config
-parameters : array
+ModeTestUnitaires : bool
+getParam(name : string, defaultValue : string) : string
-loadParameters() : array
ContextDAO
+AddContext(context : Context)
+UpdateContext(context : Context)
+DeleteContext(id : int)
+ListContext() : Context[]
+GetContext(id : int) : Context
ProjectDAO
+AddProject(project : Project)
+UpdateProject(project : Project)
+DeleteProject(id : int)
+ListProjects() : Project[]
+GetProject(id : int) : Project
SectionDAO
+AddSection(section : Section)
+UpdateSection(section : Section)
+DeleteSection(id : int)
+ListSection() : Section[]
+GetSection(id : int) : Section
UserDAO
+AddUser(user : User)
+UpdatePassword(user : User)
+Connexion(user : User) : bool
TaskDAO
+AddTask(task : Task)
+UpdateTask(task : Task)
+DeleteTask(id : int)
+ListTask(display : string, sort : string, section : string) : Task[]
+GetTask(id : int) : Task
Interface
I_ContextDAO
+AddContext(context : Context)
+UpdateContext(context : Context)
+DeleteContext(id : int)
+ListContext() : Context[]
+GetContext(id : int) : Context
I_ProjectDAO
+AddProject(project : Project)
+UpdateProject(project : Project)
+DeleteProject(id : int)
+ListProjects() : Project[]
+GetProject(id : int) : Project
I_SectionDAO
+AddSection(section : Section)
+UpdateSection(section : Section)
+DeleteSection(id : int)
+ListSection() : Section[]
+GetSection(id : int) : Section
I_UserDAO
+AddUser(user : User)
+UpdatePassword(user : User)
+Connexion(user : User) : bool
I_TaskDAO
+AddTask(task : Task)
+UpdateTask(task : Task)
+DeleteTask(id : int)
+ListTask(display : string, sort : string, section : string) : Task[]
+GetTask(id : int) : Task




Visual Paradigm Standard(tdl946724@IUT DIJON))
Interface
I_UserService
+Connexion(username : string, password : string) : bool
+ChangePassword(username : string, password : string, newPassword : string)
+GetId(login : string, password : string) : int
+Register(login : string, password1 : string, password2 : string)
+Delete(id : int, password : string)
UserService
+Connexion(username : string, password : string) : bool
+ChangePassword(username : string, password : string, newPassword : string)
+GetId(login : string, password : string) : int
+Register(login : string, password1 : string, password2 : string)
+Delete(id : int, password : string)
I_ProjectService
+AddProject(id : int, title : string)
+UpdateProject(id : int, title : string)
+DeleteProject(id : int)
+ListProject() : Project[]
+GetProject(id : int) : Project
ProjectService
+AddProject(id : int, title : string)
+UpdateProject(id : int, title : string)
+DeleteProject(id : int)
+ListProject() : Project[]
+GetProject(id : int) : Project
I_SectionService
+AddSection(id : int, title : string)
+UpdateSection(id : int, title : string)
+DeleteSection(id : int)
+ListSection() : Section[]
+GetSection(id : int) : Section
SectionService
+AddSection(id : int, title : string)
+UpdateSection(id : int, title : string)
+DeleteSection(id : int)
+ListSection() : Section[]
+GetSection(id : int) : Section
ContextService2
+AddContext(id : int, title : string)
+UpdateContext(id : int, title : string)
+DeleteContext(id : int)
+ListContext() : Context[]
+GetContext(id : int) : Context
ContextService
+AddContext(id : int, title : string)
+UpdateContext(id : int, title : string)
+DeleteContext(id : int)
+ListContext() : Context[]
+GetContext(id : int) : Context
TaskService2
+AddTask(title : string, notes : string, section : string, context : string, project : string, date : DateTime, url : string)
+UpdateTask(id : int, title : string, notes : string, section : string, context : string, project : string, date : DateTime, url : string)
+DeleteTask(id : int)
+listTask(display : string, sort : string, section : string) : Task[]
+TaskDone(id : int)
+GetTask(id : int) : Task
TaskService
+AddTask(title : string, notes : string, section : string, context : string, project : string, date : DateTime, url : string)
+UpdateTask(id : int, title : string, notes : string, section : string, context : string, project : string, date : DateTime, url : string)
+DeleteTask(id : int)
+listTask(display : string, sort : string, section : string) : Task[]
+TaskDone(id : int)
+GetTask(id : int) : Task




Controllers
Pages
Context
Home
Project
Section
Task
User
Router
Route
Context
Home
Project
Section
Task
User
Services
Interface
Logic
Model
DAO
Interface
Config
View
Context
Home
Project
Section
Task
User
Utils
Public
CSS
Media
Task