|  |  |  |
| --- | --- | --- |
| Travaux Disciplinaires au CFPT | mai 18  2017 | |
| Documentation comportant les points essentiels du développement de l’application : « Travaux Disciplinaire au CFPT » | | Documentation technique |

# Introduction

## Punitions au CFPT

Les punitions au CFPT sont assez communes aux autres écoles. Déjà, celles-ci diffèrent selon la gravité de l’acte pour lequel l’élève est puni. Évidemment, pour les petits écarts de comportement la punition donnée ne sera pas la même que pour un taux d’absences élevé. Actuellement, comme type de punition nous avons :

* **Le travail disciplinaire :** C’est un travail que l’élève doit faire à la maison. C’est la punition qu’on donnera pour de petits écarts de comportements.
* **Semaine(s) de renvoi (mise à pied) :** L’élève n’a plus le droit de venir assister au cours le temps d’une ou deux semaines en fonction de la gravité des faits. C’est la punition qu’on donnera pour acte relativement grave.

Un troisième type existait aussi : **la retenue**. Avant l’année scolaire 2014-2015, ce type de punition était donné lorsque la gravité de l’acte de l’élève se trouvait entre le **travail disciplinaire** et la **mise à pied.**

## Qu’est-ce qu’un travail disciplinaire ?

Un **travail disciplinaire** est un travail, donné par le **responsable**, que doit exécuter celui qui n’a pas suivi le règlement interne. Dans la plupart des cas, celui qui doit exécuter un travail est un **élève** et le responsable est son **professeur**.

Attention à ne pas confondre **punition** et **travail disciplinaire** ! Un travail disciplinaire est une punition mais une punition n’est pas forcément un travail disciplinaire. En effet, une punition peut se donner de plusieurs façons différentes (exemple : retenue, privation de bien, etc…).

## Comment se fait un travail disciplinaire ?

Souvent, le travail disciplinaire se fait sous forme de texte à recopier. La plupart du temps, ces textes sont : des articles Wikipédia, nouvelles, pages du dictionnaire où le règlement de l’école en question.

Mais celui-ci peut se donné de manières différentes. Je peux notamment citer M. Beney, professeur suivant mon TPI, qui demandait aux élèves punis de mettre un point de couleur différente dans chaque case d’une feuille à carreaux. Enfin, je dirais que la seule limite au travail disciplinaire est l’imagination du professeur qui la donne.

## But pédagogique ? (débat)

# Étude d’opportunité

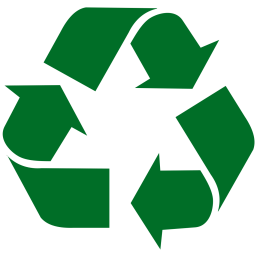
Mon application **Travaux Disciplinaires au CFPT** remplace la manière actuelle de donner un travail disciplinaire c’est-à-dire donné à l’élève un texte à recopier sur papier.

## Pourquoi ce sujet ?

Depuis toujours, le travail disciplinaire est donné par l’enseignant qui doit se charger de trouver un texte à recopier pour l’élève. Il arrive que le texte choisi ne corresponde pas au temps que devrait passer l’élève à le recopier.

De plus, il est impossible pour l’enseignant de savoir, réellement, combien de temps a pris l’élève pour recopier son texte. Ainsi si un élève affirme avoir passé le temps exigé sur son travail mais qu’il n’est pas fini, le professeur ne peut pas avoir de réelles confirmations de ces dires.

Figure 1 – Temps

Ensuite, l’exécution d’un travail peut faire gaspiller une quantité non négligeable de papier. Selon la punition, on peut compter au moins 5 feuilles de gâchées en un seul travail. Ces feuilles ne seront évidemment pas réutilisées par l’élève qui finira probablement par les jeter.

Même si aucune statistique ne concerne directement la consommation de papier via les travaux disciplinaires, la consommation de papier en général atteint des chiffres impressionnants.   
**398 millions** de tonnes de papier sont consommés chaque année dans le monde[[1]](#footnote-1). On peut donc imaginer qu’un peu « d’économie » de papier ne ferait pas de mal.

Figure - Recyclage

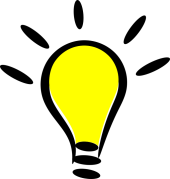
Et puis, la gestion des travaux disciplinaires peut vite devenir compliquée. L’enseignant doit vérifier si l’élève a terminé son travail en regardant si le texte du travail correspond au texte donné. Certes, le temps passé sur la vérification d’un seul travail disciplinaire n’est pas conséquent mais ce temps se multiplie par le nombre de travaux à vérifier. Ainsi, on peut imaginer le temps que passerait un enseignant à la vérification des travaux si la punition inclus toute une classe.

Figure 3 - Vérification

***\*Parler De Pédagogie ici\****

C’est pour toutes ces raisons que mon choix s’est porté sur l’application Travaux Disciplinaires au CFPT. En effet cette application permet de remédier à ces problématiques qui viennent avec le concept du travail disciplinaire. Voici les solutions à celles-ci :

Figure 4 - Solution

* La durée moyenne que devrait prendre un élève à recopier le texte sera affiché. De plus, des textes prédéfinis de 20, 40, 60, 120 et 150 minutes seront à dispositions
* La durée effective du travail de l’élève sera calculée.
* L’exécution du travail ne nécessitera pas de papier.
* La vérification de la progression du travail se fera par le programme et affiché. Ainsi ce n’est pas au professeur de comparé les deux textes afin de savoir s’il a été recopié dans les règles.
* La copie du texte entrainera l’élève à la dactylographie.

## Comparaison avec d’éventuels programmes existants

Tout d’abord, je tiens à préciser qu’il ne semble pas y avoir d’application dont le but est similaire à celui de mon application, sur internet.

Je dirais que le type d’application qui se rapproche plus de la mienne est ceux que l’on peut retrouver sur les sites vous apprenant la dactylographie. En effet, ces applications affichent un texte et demande à l’utilisateur de le retaper en filtrant les fautes, ce qui est une de mes fonctionnalités principales. Ainsi nous pouvons voir des sites comme *typingstudy.com* proposer, par ce biais, une petite formation à la dactylographie

# Analyse fonctionnelle

## Fonctionnalité globale

L’application que je dois réaliser sert à créer, exécuter et gérer des travaux disciplinaires. Celle-ci permettrait au professeur d’envoyer le travail à l’élève et à celui-ci de l’exécuter puis le rendre au professeur qui s’assurerait, via le système de gestion de travaux, que l’élève a fini son travail. Le tout se fait donc sur une seule et même application. Évidemment, l’élève peut créer lui-même son travail disciplinaire.

## Fonctionnalités détaillées

### Créer un travail disciplinaire

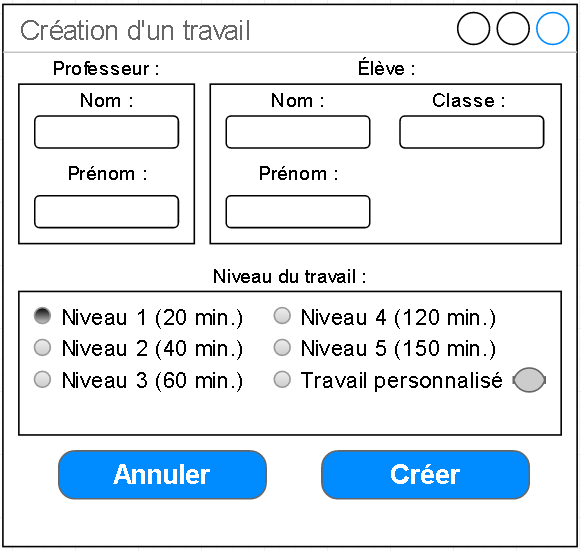
****

Figure 3 - Page de création d'un travail disciplinaire

Voici la page de création de travail disciplinaire.

Dans la case « **Professeur** » se trouvent les zones de saisie ou doivent être indiqué le nom et prénom du professeur.

Dans la case « **Élève** » se trouvent les zones de saisie ou doivent être indiqué le nom, le prénom et la classe de l’élève.

Dans la case « **Niveau du travail**» se trouvent les différents niveaux à disposition. Si le niveau choisi est travail personnalisé, l’utilisateur devra sélectionner un fichier texte en cliquant sur le bouton à coté de texte personnalisé.

5 textes prédéfinis sont à disposition. Chacun de ceux-ci correspondent à un niveau différent. Voici comment s’organise ceux-ci :

* **Niveau 1 :** environ 20 minutes
* **Niveau 2 :** environ 40 minutes
* **Niveau 3 :** environ 60 minutes
* **Niveau 4 :** environ 120 minutes
* **Niveau 5 :** environ 150 minutes

Si l’utilisateur clique sur le bouton « **Créer** », un travail se crée avec les données tel quel et s’ajoute à la liste de travaux disciplinaires.

En revanche, s’il clique sur le bouton « **Annuler** », cela annule l’action et ferme la page de création.

### Exécution du travail

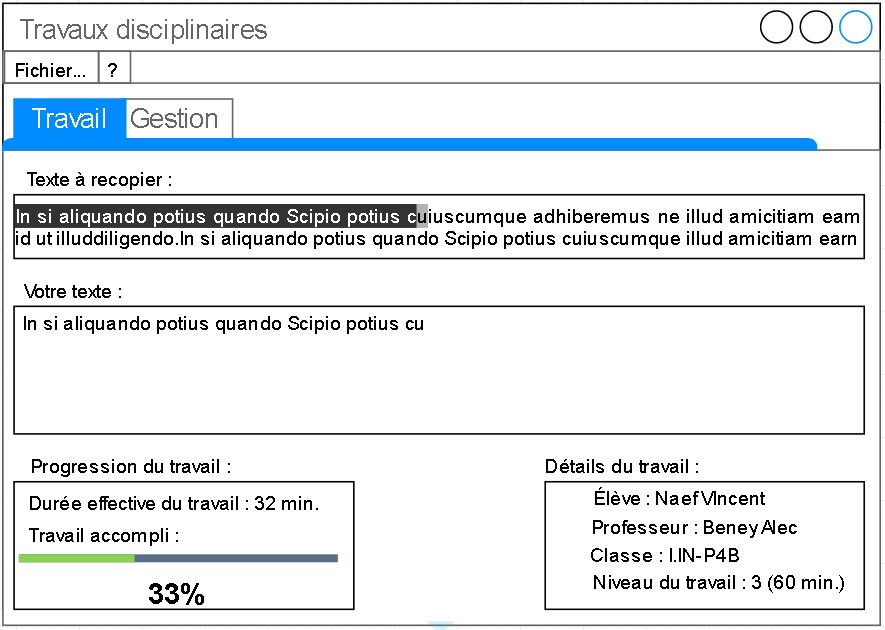


Figure 4 - Onglet d'exécution du travail

Voici l’onglet dans lequel l’élève chargé du travail disciplinaire doit effectuer son labeur.

Dans la zone de saisie « **Texte à recopier** » se trouve le texte que l’utilisateur doit recopier Le texte vêtu d’un fond gris est celui qui est déjà copié par l’utilisateur.

Dans la zone de saisie « **Votre texte** » se trouve le texte copié par l’élève.

Dans la case « **progression du travail** » se trouve la durée que l’élève a passé sur son travail ainsi que la progression de celui-ci.

Dans la case « **Détails du travail** » se trouve les données « statiques » du travail, c’est-à-dire :

* Le nom, prénom et la classe de l’élève
* Le nom et prénom du professeur
* Le niveau et la durée estimée du travail.

Lorsque l’utilisateur recopie le texte affiché, l’application veille à ce que celui-ci ne puisse pas taper de mauvais caractères. Si celui-ci ne trouve pas la façon d’entrer un caractère, l’application lui affiche la solution.

De plus, elle calcule le temps que l’utilisateur passe sur son travail. S’il arrive que l’utilisateur ne tape rien pendant plus de 5 secondes, le chronomètre s’arrête. Celui-ci reprend lorsque l’utilisateur commence à retaper.

Enfin, elle calcule et affiche la progression du travail. En plus d’afficher le nombre de caractères tapés sur le nombre de caractères totaux, elle affiche aussi en pourcentage l’avancée du travail.

### Gestion des travaux

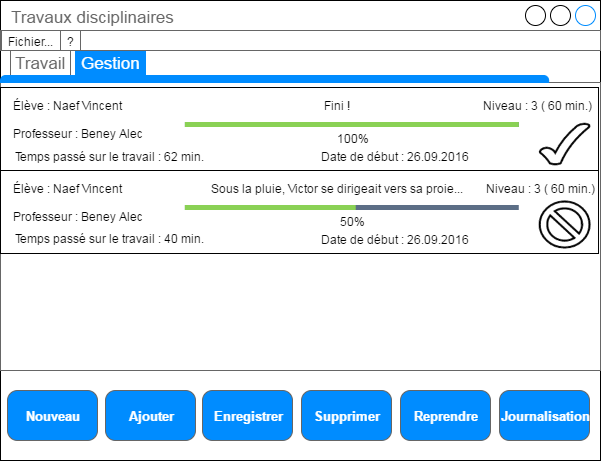
****

Figure 5 - Onglet de gestion de travail

Voici l’onglet de gestion de travaux. Dans cet onglet est affichée une liste des travaux que l’utilisateur a ouvert ou crée durant la même session. Dans cette liste de travaux, chaque travail est affiché comme ceci :

* Nom, prénom et classe de l’élève
* Nom et prénom du professeur
* Le temps du travail
* Le niveau du texte
* La progression
* Une image indiquant l’état du travail

#### Fonctionnalités des boutons

Les boutons en dessous de la liste de travaux permettent d’interagir avec celle-ci. Voici leurs fonctionnalités :

* Le bouton « **Nouveau** » permet de créer un travail qui s’ajoutera à la liste de travaux.
* Le bouton « **Ajouter** » permet d’ajouter un travail contenu dans un fichier .td
* Le bouton « **Enregistrer** » permet d’enregistrer le travail sélectionné
* Le bouton « **Supprimer** » permet de supprimer le travail sélectionné de la liste
* Le bouton « **Reprendre** » permet à l’utilisateur de continuer la progression du travail sélectionné.
* Le bouton « **Journalisation** » permet de créer un fichier texte contenant toutes les données importantes de la liste de travaux.

# Analyse organique

## Description globale

Comme indiqué dans le cahier des charges, le programme est codé en MV (Modèle-Vue). Le MV reprend le principe du MVC (Modèle-Vue-Controller) sauf que le « controller » est inclus dans le code de la vue.

\*Inclure diagramme de classe\*

## Explication des méthodes utilisées

### Classe TravailDisciplinaire :

#### Champs / Propriétés

* **DateDeDebut :** Date de début du projet.
* **Professeur :** Nom et prénom du professeur.
* **Eleve :** Nom, prénom et classe de l’élève.
* **CleValidation :** Objet « TravailDisciplinaire » encodé en MD5. Sert à vérifier si le travail a été modifié sans l’aide du programme « Travaux Disciplinaires au CFPT ».
* **Progression :** Progression du travail (nombre de caractères tapés)
* **Niveau :** Niveau et texte du travail
* **DureeEffective :** Temps que l’utilisateur a passé sur son travail

#### Constructeur TravailDisciplinaire

*Paramètres :*

* String paramNomProf
* String paramPrenomProf
* String paramNomEleve
* String paramPrenomEleve
* String paramClasse
* Niveau paramNiveau

Créer un travail disciplinaire et initialise les valeurs :

* **Élève :** Nom, prénom et classe de l’élève.
* **Professeur :** Nom et prénom du professeur.
* **Niveau :** Niveau du texte et texte à recopier.
* **Progression :** 0 (car il vient d’être créé).
* **Temps :** 0 (c’est-à-dire 0 heure et 0 seconde).
* **Date de début :** Date du moment de la création du travail.

#### Méthode string ProgressionToString

Retourne sous forme de texte la progression sous ce format : « xxx caractère(s) sur yyy ».

#### Méthode bool VerifierCaractere

*Paramètre :*

* *char paramCaractere*

*Vérifie si le caractère passé dans paramCaractere est le caractère que l’utilisateur devait taper.* Détermine que paramCaractere est le bon caractère s’il est égal à celui qu’il faut taper **\*Structograme\***

Retourne si oui ou non le caractère tapé est le bon.

#### Méthode void AvancerProgression

Avance la progression de 1.

#### Méthode bool EstFini

*Vérifie si le travail est fini.* Détermine que le travail est fini si la progression est égale au nombre de caractère du texte à recopier.

Retourne si oui ou non le travail est fini.

#### Méthode int CompterCaracteres

Retourne le nombre de caractères que contient le texte à recopier.

#### Méthode int CalculerPourcentageEffectue

*Calcule le pourcentage effectué du travail sur 100.* Détermine ce pourcentage en divisant le nombre de caractère tapé (progression) par le nombre de caractère total et multiplie le tout par 100.

Retourne le pourcentage effectué du travail.

#### Méthode string GetTexteTapeParUtilisateur

*Va chercher le texte déjà tapé par l’utilisateur.* Détermine ce texte en allant chercher tous les caractères du texte à recopier pour autant que l’index de ceux-ci soit inférieur à la progression.

Retourne le texte déjà tapé par l’utilisateur.

#### Méthode string MinuteEtSecondeToString

*Transforme en texte le nombre de minute et de seconde passé sur le travail par l’utilisateur.* Calcule le nombre de minute et de seconde avec les ticks des champs « DureeEffective ». \*Insérer structogramme ici\*

Retourne le nombre de minute et de seconde en texte sous ce format : « xxx min. et yyy sec. »

#### Méthode string NiveauToString

Retourne le Texte que retourne le ToString de niveau

#### Override string ToString

Retourne sous forme de texte les informations importante de la classe TravailDisciplinaire (donc de lui-même)

#### void SerialiserTravail

*Sauvegarde le travail disciplinaire.* Appel la méthode « CryptageTravail » et créer un fichier à l’endroit passé en paramètre est sérialise en binaire dedans l’objet travail disciplinaire (lui-même)

#### TravailDisciplinaire DesierialiserTravail

*Paramètre*

* string paramFichier

*Déserialise le travail disciplinaire.* Lis les données du fichier paramFichier et retourne le travail avec ces données s’il n’y a pas d’erreur. S’il y a une erreur, envoie l’exception au parent.

Retourne un travail disciplinaire.

#### void CryptageTravail

*Stock le travail crypter dans CleValidation.* Crypte les données retourné par le ToString du travail en md5 est stock le résultat dans le champs CleValidation .

#### VerifierDonneeTravail

*Vérifie si le travail a été modifié sans cette l’application Travaux Disciplinaire au CFPT.* Si la méthode CryptageTravail retourne autre chose que la valeur stocké dans le champs CleValidation.

Retourne si oui ou non le travail est valide.

### Classe Niveau :

#### Constructeur Niveau

*Paramètres :*

* int paramNiveau
* string paramTexte

Créer et initialise les valeurs :

* **NumeroNiveau  :** Niveau du travail.
* **TexteARecopier :** Texte à recopier du travail

#### int CompterCaractere

Retourne le nombre de caractère du TexteARecopie

#### Int CalculerMinuteDuTexte

*Calcule le temps approximatif que prendrait le texte à être écrit.*

Divise le nombre de caractères du texte par 5 pour donner le nombre de mot. *(En moyenne un mot est constitué de 5 caractères)*

Divise le nombre de mots par 33 pour donner le nombre de minutes *(En moyenne, une personne tape 33 mots par minutes)*

Arrondit le résultat en un multiple de 10.

Retourne le nombre de minutes (arrondi à 10) estimés pour recopier le texte.

#### Override string ToString

Retourne en chaine de caractères l’objet niveau sous ce format :   
« \*numéroNiveau \* (~ \*nbMinutes\* min.) »

Exemple : «  3 (~60 min.)  »

#### String FiltrerCaracteres

*Paramètre :*

* String paramTexteAFiltrer

*Supprime les retours à la ligne en trop et le \r d’Environement.NewLine.*

*(À chaque retour à la ligne se trouve les deux caractères \r\n. Le \r ne sert qu’à la mise en forme pour les fichier « .docx » par exemple.)*

La fonction va d’abord remplacer les « Environment.NewLine » qui se trouvent à la suite par un seul « Environment.NewLine ». Ensuite il va supprimer tous les caractères \r qu’il trouve dans le paramTexteAFiltrer.

*Retourne paramTexteAFiltrer (sans retour à la ligne de trop et sans caractère «  \r »*

### Classe Personne

#### Champs / Propriétés

* **Prenom :** Prénom de la personne
* **Nom :** Nom de la personne

#### Constructeur Personne

*Paramètres :*

* String paramNom
* String paramPrenom

Créer un travail disciplinaire et initialise les valeurs :

* **Nom :** paramNom
* **Prenom :** paramPrenom

#### Override ToString

Retourne le nom et prénom de la personne sous format texte.

### Classe Eleve

Descend de la classe Personne.

#### Champ / Propriété

* **Classe :** Classe de l’élève

#### Constructeur Eleve

*Paramètres :*

* String paramNom
* String paramPrenom
* String paramClasse

Créer un travail disciplinaire et initialise les valeurs :

* **Nom :** paramNom
* **Prenom :** paramPrenom
* **Classe :** paramClasse

### Form frmCreation

### Form frmPrincipale

## Méthodes de réalisation

### Créer un travail

# Tests et protocole de tests

# Conclusion

# Bibliographie

**Figure 1 – Recyclage** https://pixabay.com/fr/recyclage-symbole-logo-vert-%C3%A9co-304974/

**Figure 2 – Vérification** https://www.alesiacom.com/services/gestion-de-projets

**Figure 3 – Idée** <https://pixabay.com/fr/dessin-anim%C3%A9-ic%C3%B4ne-ampoule-symbole-1294877/>

**Figure 1 – Chronomètre** http://www.flaticon.com/free-icon/stopwatch\_138125#term=chronometer&page=1&position=66

1. Source : http://www.planetoscope.com/developpement-durable/papier [↑](#footnote-ref-1)