Documentation technique

Ce document est destiné à présenter les choix techniques qui permettent de comprendre l'organisation du code de cette application.
La maintenance de l'application doit se faire en respectant ces choix.

Table des matières

I - Contexte

La direction du CSE souhaite avoir un site pour gérer les cadeaux de noël de chaque salarié :

- Chaque salarié s'inscrit et peut inscrire ses enfants
- Il peut ensuite choisir son cadeau ou ceux de ses enfants
- L'administrateur peut gérer les salariés et générer un pdf récapitulatif des commandes

II - Mise en œuvre

II.1 Configuration requise pour le serveur d'application

- Serveur web : Apache + PHP 7.3 (notamment pour supporter le typage des paramètres des méthodes).
- SGBD : MySql 8.0 avec InnoDB

II.2 Procédure d'installation sur la plate-forme de développement

- Le projet est développé depuis un répertoire distinct de celui de la racine du serveur web. Il faut donc paramétrer correctement le projet NetBeans à cet effet.

- La base de données doit être créée à l'aide des fichiers de commandes SQL inclus dans le code du projet (sql`/*.sql`):
  - `sql/Script_create_bdd_ce.sql` : crée la base de données, un utilisateur spécifique avec des droits limités à cette base, ainsi que la structure des tables
  - `sql/Script_insert_bdd_ce.sql` : insère les données du jeu d'essai initial dans les tables.

- Adapter si nécessaire les valeurs des paramètres précisées dans le fichier `includes/parametres.ini`, notamment celles figurant en gras :

```
;-----------------------------------------------------------------
; paramétrage de l'application
;-----------------------------------------------------------------
[IDENTIFICATION]
version = '2021'                                ; version de l'application
auteur = 'MENADIER Mélodie, BARILLET Tom'       ; nom des auteurs du projet

[BDD_MYSQL_LOCAL]
dsn = 'mysql:host=localhost;dbname='      ; url du serveur de bases de données
bdd = bdd_ce                              ; nom de la base de données
login = util_ce         ; login d'un utilisateur de MySql avec des droits sur la BDD
mdp = ww7OivKSLNMlr1gO                    ;  mot de passe de cet utilisateur
```

III - Normes de codage

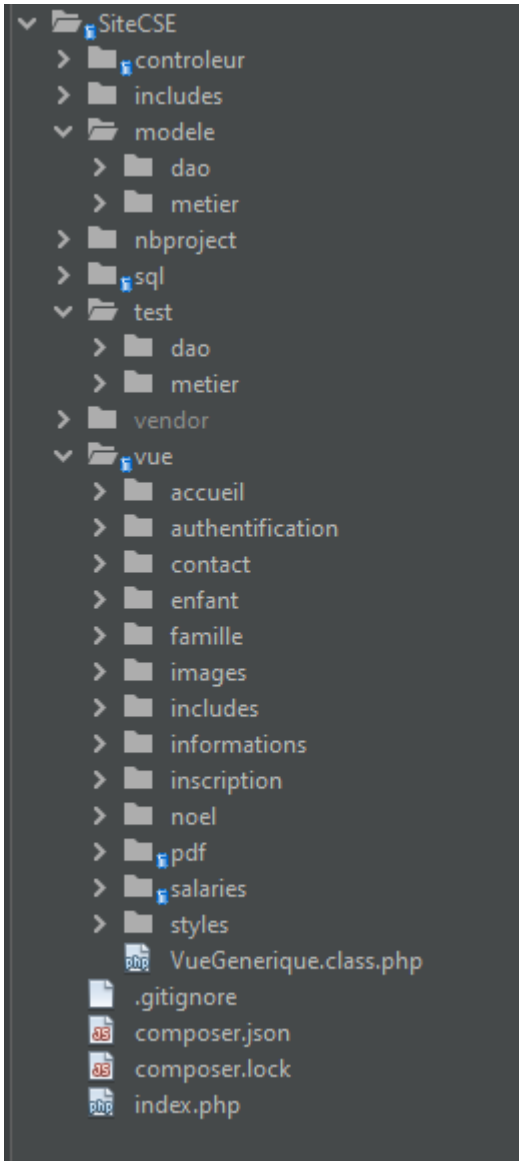
Le document de référence est celui de la section SIO du lycée La Joliverie :

`1SIO_PHP_NormesDePresentationDuCodeSource_v2018`.

On y trouve notamment des exigences en matière de commentaires, d'imbrication du code PHP et du code HTML et de nommage des identificateurs.

IV - Structure du code (MVC)

Le code est organisé selon le patron de conception Modèle / Vue / Contrôleur. Voici l'arborescence de répertoires :



/controleur, contient :

- la classe *ControleurGenerique.class.php* dont héritent toutes les classes contrôleurs
- les classes de contrôle des vues, préfixées *Ctrl* (exemple : *CtrlAccueil.class.php*)
- les autres classes de gestion (exemple : *GestionErreurs.class.php*)

/includes, contient les fichiers à inclure dans les contrôleurs (exemples : *fonctionsUtilitaires.inc.php*, *parametres.ini*)

/modele, contient :

- les classes métier (exemple : *Accueil.class.php*)
- les classes DAO, suffixées *DAO* (exemple : *AccueilDAO.class.php*)

/sql, contient les fichiers de commande SQL nécessaires à la création de la base de données.

/test, contient les programmes de tests unitaires et d'intégration, suffixés *Test* (exemple : *AccueilTest.php*).

/vue, contient :

- la classe *VueGenerique.class.php* dont héritent toutes les classes de vues.
- un répertoire par module contenant les classes de vues, préfixées *Vue* (exemple : ***accueil**/VueAccueil.class.php*)
- le répertoire ***images*** contient les fichiers images à inclure (ex. : *logo.png*)
- le répertoire ***includes*** contient les fichiers (ex. : *onglets.inc.php*) à inclure dans le code HTML généré par les classes de vues (méthode *afficher*). N.B. :
  - Le fichier *debut.inc.php* contient le début du code HTML commun à toutes les pages du site (entête, bouton de connexion, onglets)
  - Le fichier *fin.inc.php* contient la fin du code HTML commune à toutes les pages du site.
- le répertoire ***javascript*** contient les fichiers de code javascript utilisés par les vues (ex. : *datePicker.js*).
- le répertoire ***styles*** contient les fichiers de feuilles de style à associer au code HTML des vues (ex. : *cssOnglets.css*).

/index.php est le contrôleur frontal, chargé de décoder les requêtes HTTP pour invoquer la méthode du contrôleur adaptée.
Exemple d'URL : http://localhost/ProjetSiteCSE/projet-cse-stage/cse-stage/index.php?controleur=enfant&action=defaut
provoquera l'instanciation d'un objet de la classe CtrlEnfant : `$ctrl = new CtrlEnfant ();`
puis un appel de la méthode créer de cet objet : `$ctrl->creer();`

/vendor est un répertoire qui correspond à la classe html2pdf qui permet de génére run pdf à partir d'une page html.
Voici le lien du git pour l'installer et avoir accès à la documentation : https://github.com/spipu/html2pdf

V - Programmation orientée objets

V.1 Espaces de noms

V.1.1 - Déclaration

Chaque classe doit déclarer son espace de nom à l'aide du mot-clef *namespace* ; par convention dans ce projet, les espaces de noms correspondent au sous répertoire du projet contenant la classe.
Exemple : le fichier *Accueil.class.php* est situé dans le répertoire *modele\metier*

```
<?php namespace modele\metier;
	class Accueil {
	    [...]
	}
```

V.1.2 - Utilisation

Il faut déclarer l'utilisation d'une classe avec le mot-clef *use*.
Exemple : la classe AccueilDAO utilise la classe Accueil.

```
<?php namespace modele\dao;
	use modele\metier\Accueil;
	use modele\metier\Contact;
	use modele\metier\Enfant;
	use PDO;
	class AccueilDAO
	{
		[...]
	}
```

V.1.3 - Chargement automatique

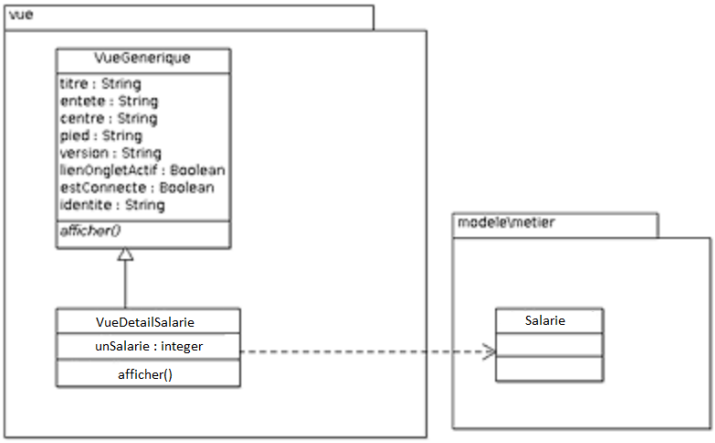
En PHP, le code d'une classe doit être inclus dans le code qui l'utilise. Dans ce projet, cette inclusion est automatisée grâce à la fonction « spl_autoload_register » (fichier *includes/autoload.inc.php*).

```
<?php
	function chargeurBaseEspacesDeNoms($className) {
	//    $className = ltrim($className, '\\');
	    $fileName  = __DIR__.'/../';

	    $fileName  .= str_replace('\\', DIRECTORY_SEPARATOR, $className).'.class.php';
	       if (file_exists($fileName)) {
	       require_once($fileName);
	    } else {
	        throw new Exception("Autoload - problème de chargement de la classe $className - Le
	fichier : " . $fileName . " n\'existe pas.");
	    }

	}

	// ajout du chargeur ppour un chargement automatique
	spl_autoload_register("chargeurBaseEspacesDeNoms");
```

V.2 Classes MVC

V.2.1 - Vues

Chaque classe du paquetage *vue* hérite de la classe *VueGenerique* (*VueGenerique.class.php)* et doit donc implémenter la méthode abstraite *afficher* de cette dernière.



La méthode afficher a pour rôle de produire le code HTML de la page visée, en y intégrant les données mémorisées dans les attributs de la classe.

La méthode *afficher* :

- commence par l'inclusion du fichier *vue/includes/debut.inc.php* (titre, bouton de connexion, onglets)
- ajoute la code permettant d'afficher le centre de la page
- termine par l'inclusion du fichier *vue/includes/fin.inc.php*

Les attributs de la classe *VueGenerique* et de sa sous-classe doivent tous être initialisés par le contrôleur avant d'appeler la méthode afficher, sauf *entete* et *pied* qui le sont par le constructeur de *VueGenerique*.

Attributs de la classe *VueGenerique* :

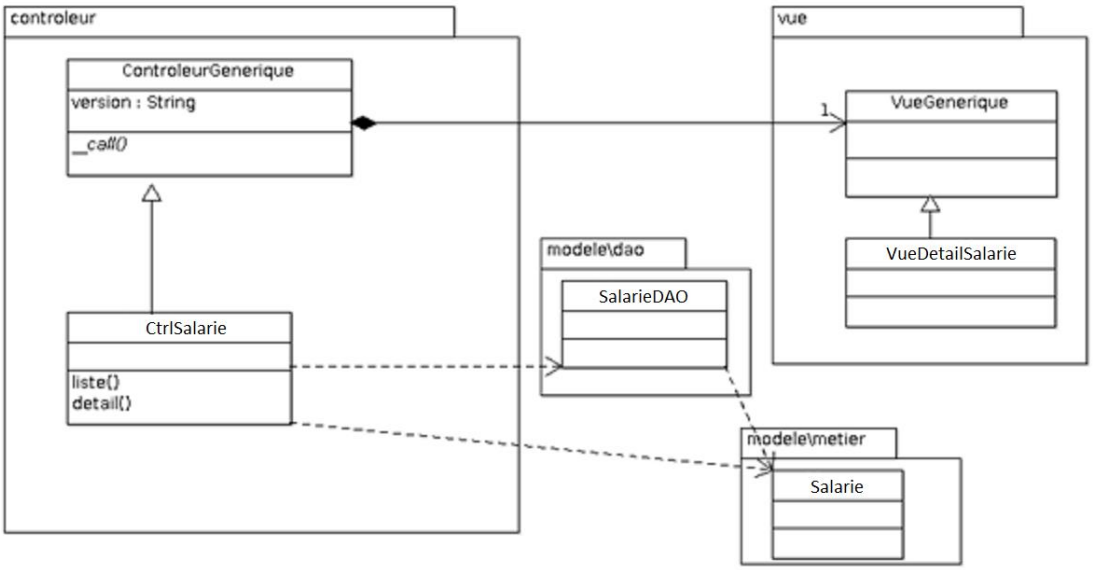
```
<?php
[...]
 abstract class VueGenerique {
    /** @var string titre de la vue (dans debut.inc.php) */
    private $titre;
    /** @var string chemin d'accès vers le fichier d'inclusion pour l'entête  */
    private $entete;
    /** @var string chemin d'accès vers le fichier d'inclusion pour la partie centrale */
    private $centre;
    /** @var string chemin d'accès vers le fichier d'inclusion pour le pied de page */
    private $pied;
    /** @var bool statut des liens des onglets ; =true => actif ; =false => inactif (dans
debut.inc.php) */
    private $lienOngletActif;
    /** @var bool un utilisateur est-il authentifié sur cette session */
    private $estConnecte;
    /** @var string identité de l'utilisateur connecté (à afficher à coté du bouton) */
    private $identite;

    public function __construct() {
        // initialisation des valeurs par défaut
        $this->setTitre('Site CSE');
        $this->setEntete(GestionParametres::racine() . 'vue/includes/debut.inc.php');
        $this->setPied(GestionParametres::racine() . 'vue/includes/fin.inc.php');
    }

    /** Afficher la vue signifie l'inclure au flux de sortie HTML */
    public abstract function afficher();
    [...]
}
```

Exemple de sous-classe, *VueDetailSalarie* :

```
<?php
 [...]
class VueDetailSalarie extends VueGenerique {
    /** @var Salarie identificateur du salarie à afficher */
    private $unSalarie;
     […]
}
```

V.2.2 - Contrôleurs particuliers

Chaque classe contrôleur hérite de la classe *ControleurGenerique* (*ControleurGenerique.class.php*).



Un contrôleur particulier présente autant de méthodes statiques qu'il y a d'actions possibles sur un module. Par exemple, pour le module de gestion des salariés, le contrôleur présente les méthodes liste, détail, créer, validerCreer, modifier, validerModifier, supprimer, validerSupprimer.
Chaque méthode d'action est chargée :

- d'instancier la vue correspondante,
- de lui fournir les données nécessaires en renseignant ses attributs,
- puis lui demander de produire le code HTML ($this->vue->afficher())

Extrait du contrôleur générique :

```php
<?php
abstract class ControleurGenerique {
    /** @var VueGenerique Association OneToOne Controleur -> Vue */
    protected $vue;
    /** @var string identification de la version de l'application à afficher */
    protected $version;
    /**
* Action par défaut. Devra être implémentée par chaque contrôleur
     */     public abstract function defaut()
```

Extrait d'un contrôleur particulier :

```php
<?php
class CtrlSalaries extends ControleurGenerique {
    /** controleur= salarie & action= defaut
       * par défaut, afficher la liste des salaries  */
    public function defaut() {
       $this->liste();     }
       [...]
    /** controleur= salaries & action=detail & id=identifiant_salarie
* Afficher un salarie d'après son identifiant     */
public function detail() {
       $idSalarie = $_GET["id"];
        $this->vue = new VueDetailSalarie();
        // Lire dans la BDD les données du salarié à afficher
        Bdd::connecter();
        $this->vue->setUnSalarie(SalarieDAO::getOneById($idSalarie));
       $this->vue->setEnfants(EnfantDAO::getAllById($idSalarie));
        parent::controlerVueAdmin();
        $this->vue->setTitre("CSE La Joliverie - Salariés");
        $this->vue->afficher();
    }
[...]
```

V.2.3 - Contrôleur frontal (index.php)

C'est le rôle du fichier *index.php*. Ce n'est pas une classe, mais un simple programme PHP.
Il doit analyser l'URL de la requête HTTP pour déterminer quel contrôleur instancier, et quelle méthode du contrôleur appeler (action).

Exemple : http://localhost/ProjetSiteCSE/projet-ces-stage/cse-stage/index.php?controleur=salaries&action=modifier&id=1

Ici, le contrôleur frontal instanciera un objet *Salarié* et appellera sa méthode *detail()* . Le paramètre « id » sera exploité par la méthode *detail()*...

```php
<?php
[...]
    // analyse de l'URL (paramètres GET)
    $controleur = $_GET['controleur']; b
    $action = $_GET['action'];
[...]
    // Construction du nom de la classe contrôleur
    $classeControleur = "controleur\Ctrl" . ucfirst($controleur);
    // Instanciation d'un obje contrôleur
    $ctrl = new $classeControleur();
    // Appel de la méthode d'action de ce contrôleur
    $ctrl->$action();
[...]
```

V.3 Classes de gestion

V.3.1 - Gestion des sessions

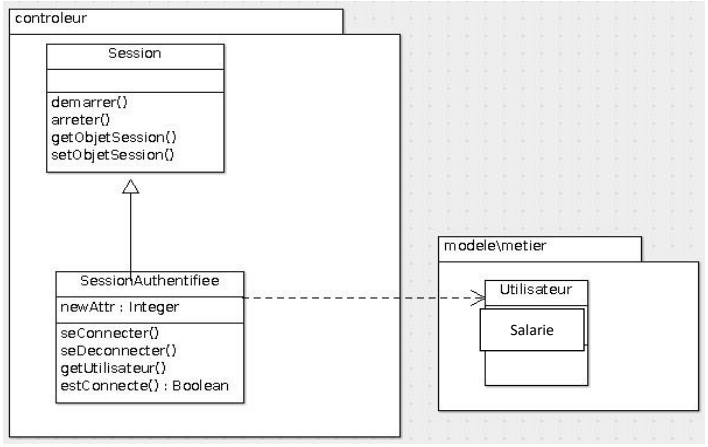
L'utilisation des sessions PHP est regroupée dans deux classes comportant des méthodes statiques :

V.3.1.1 Session

La classe *Session* (controleur/) regroupe le code permettant la gestion des sessions PHP.
Elle est utilisée par les classes *GestionParametres* et *GestionErreurs*.

V.3.1.2 Session authentifiée

La classe *SessionAuthentifiee* (controleur/) permet de contrôler et de mémoriser l'authentification d'un utilisateur (Salarie) de l'application à l'aide de la session PHP. *SessionAuthentifiee* hérite de *Session.*



Vérification de l'authentification

Dans le contrôleur frontal (index .php) on trouve cet extrait de code :

```
// Contrôle de l'authentification
// seules les pages d'accueil, d'authentification, et de contact sont accessible hors connexion
if (!SessionAuthentifiee::estConnecte() && $controleur != 'accueil'&& $controleur != 'authentification') {
    $controleur = 'accueil';
    $action = 'refuser';
}
```

Validation d'une authentification

Dans la méthode authentifier() du contrôleur CtrlAuthentification, on trouve le code suivant :

```
<?php
    /** controleur= authentification & action= authentifier
     * vérifier l'identité de l'utilisateur dans la BDD    */
    public function authentifier() {
        Bdd::connecter();
        $mail = $_REQUEST['adressemail'];
        $mdp = $_REQUEST['motdepasse'];
        $utilisateurConnecte = $this->verifierDonneesIdentification($mail, $mdp);
        if (GestionErreurs::nbErreurs() == 0) {
            //L'utilisateur est authentifiée
            // Initialiser la session
            SessionAuthentifiee::seConnecter($utilisateurConnecte);
           // Afficher l'écran d'accueil
header("Location: index.php"); [...]
```

V.3.2 - Gestion des paramètres

La classe GestionParametres sert à lire le fichier includes/parametres.ini et à initialiser un tableau 'parametres' dans la session PHP avec ces valeurs.
Cela permet de contrôler l'installation de l'application sans en modifier le code.

```
;-------------------------------------------------------------------
; paramétrage de l'application
;-------------------------------------------------------------------
[IDENTIFICATION]
version = '2021'                                    ; version de l'application
auteur = 'MENADIER Mélodie, BARILLET Tom'           ; nom des auteurs du projet

[BDD_MYSQL_LOCAL]
dsn = 'mysql:host=localhost;dbname='      ; url du serveur de bases de données
bdd = bdd_ce                              ; nom de la base de données
login = util_ce          ; login d'un utilisateur de MySql avec des droits sur la BDD
mdp = ww7OivKSLNMlr1gO                    ;  mot de passe de cet utilisateur
```

V.3.3 - Gestion des erreurs

Deux mécanismes sont à l'œuvre pour gérer les erreurs.

V.3.3.1 Gestion des «logs»

Il s'agit de mémoriser temporairement les erreurs détectées (à l'analyse des formulaires de saisie par exemple) afin de différer leur affichage.
Les message d'erreur sont mémorisés dans un tableau 'erreurs' de la session PHP.
La classe *GestionErreurs* offre les méthodes nécessaires.
Exemple d'utilisation :

- Extrait de code de vérification de la saisie d'un formulaire *CtrlSalaries ::verifierDonneesSalarie* :

```php
// Vérification des champs obligatoires.
// Dans le cas d'une création, on vérifie aussi l'id
if ($creation && $unSalarie->getPrenom() == "" || $unSalarie->getNom() == ""||
      $unSalarie->getEmail() == ""|| $unSalarie->getMdp() == "") {
    GestionErreurs::ajouter('Chaque champ suivi du caractère * est obligatoire');
}
```

- Affichage des erreurs ainsi enregistrées *vue/includes/fin.inc.php* :

```php
<?php
//Affichage des erreurs puis réinitialisation
echo \controleur\GestionErreurs::printErreurs();
\controleur\GestionErreurs::razErreurs();
```

V.3.3.2 Traitement des exceptions

Lorsqu'on attrape une exception, on peut enregistrer le message d'erreur à l'aide de la classe GestionErreurs afin de ne pas interrompre brutalement l'application.
Exemple détection d'un contrôleur ou d'une action inexistant.e.s dans *index.php* :

```php
[...]
// Instanciation du contrôleur et appel de sa méthode d'action
try {
    // Construction du nom de la classe contrôleur
    $classeControleur = "controleur\Ctrl" . ucfirst($controleur);
    // Instanciation d'un obje contrôleur
    /* @var controleur\ControleurGenerique $ctrl  */
    $ctrl = new $classeControleur();
    $ctrl->setVersion("v. 2018-2019");
    // Appel de la méthode d'action de ce contrôleur
    $ctrl->$action();
} catch (\Exception $e) {
    // en cas d'erreur (contrôleur ou action inexistante)
    GestionErreurs::ajouter("Module indisponible :$controleur:- " . $e->getMessage());
    header("Location: index.php");
}
[...]
```