BOUVEYRON
Nicolas

SIO 2-SLAM

Questionnaire du Package Pré-Paramétré

Période de stage du 04/03/2019 au 12/04/2019

SOMMAIRE

1. PRÉSENTATION DE L'ENTREPRISE :

ASYS est un éditeur de solutions RH pour la gestion des temps et des activités et la planification.
Pour cela, deux solutions sont proposées :

- **So'Horsys** pensée pour les PME
- **Chronos** pour les grandes entreprises et les établissements publics

So'Horsys est un logiciel web qui propose de nombreux modules RH, sont point fort est que le client peut décider d'activer les modules dont il a besoin et ainsi obtenir une application sur mesure.

Il fonctionne sur le principe du SaaS (Software as a Service).

Cette application est destinée aux petites entreprises qui nécessite d'avoir un suivi des temps, de planification des congés, données comptable etc …

Grâce à son interface intuitive et épuré, le temps de formation des utilisateurs est moindre.

Elle est accessible depuis n'importe quel support, une connexion internet est requise.

Pour développer ses applications, la société utilise leur propre framework pour ainsi uniformiser le code et assurer une évolution permanente des solutions.

Elle utilise l'environnement de développement WebDev ainsi que l'outil Visual Studio Code.

2. EXPRESSION DES BESOINS

2.1 Contexte

Dans le cadre de l'intégration des solutions applicatives de ASYS, un questionnaire à choix multiples sera proposé aux futurs clients pour déterminer les modules qu'ils souhaitent intégrés à leur solution.

L'objectif est donc de créer une interface permettant aux clients de remplir ce questionnaire mais aussi de créer une interface permettant aux intégrateurs d'ajouter, modifier un questionnaire ainsi que de visualiser l'avancement des clients.

Ce projet sera réalisé en équipe :

- BOUVEYRON Nicolas - BTS SIO 2 SLAM
- BERNARD Mathis - BTS SIO 2 SLAM

2.2 Demandeurs

Cette application sera utilisée en interne par les consultants pour faciliter l'implantation, le paramétrage des solutions auprès des clients de ASYS.

ASYS est un éditeur de solutions RH pour la gestion des temps et des activités et la planification.

2.3 Description de la demande, objectifs

Création de deux interfaces responsive design.
La première permettant de visualiser et remplir un questionnaire sera proposée aux clients.
La seconde permettant de visualiser l'avancement d'un client dans le questionnaire ainsi que la création, modification d'un questionnaire
Cette application a pour objectifs d'optimiser l'intégration de So'Horsys en apportant une rapidité d'implantation et de paramétrage.
So'Horsys est une solution RH développée par ASYS proposant de nombreux modules permettant la gestion, la planification de temps, comptabilité, …

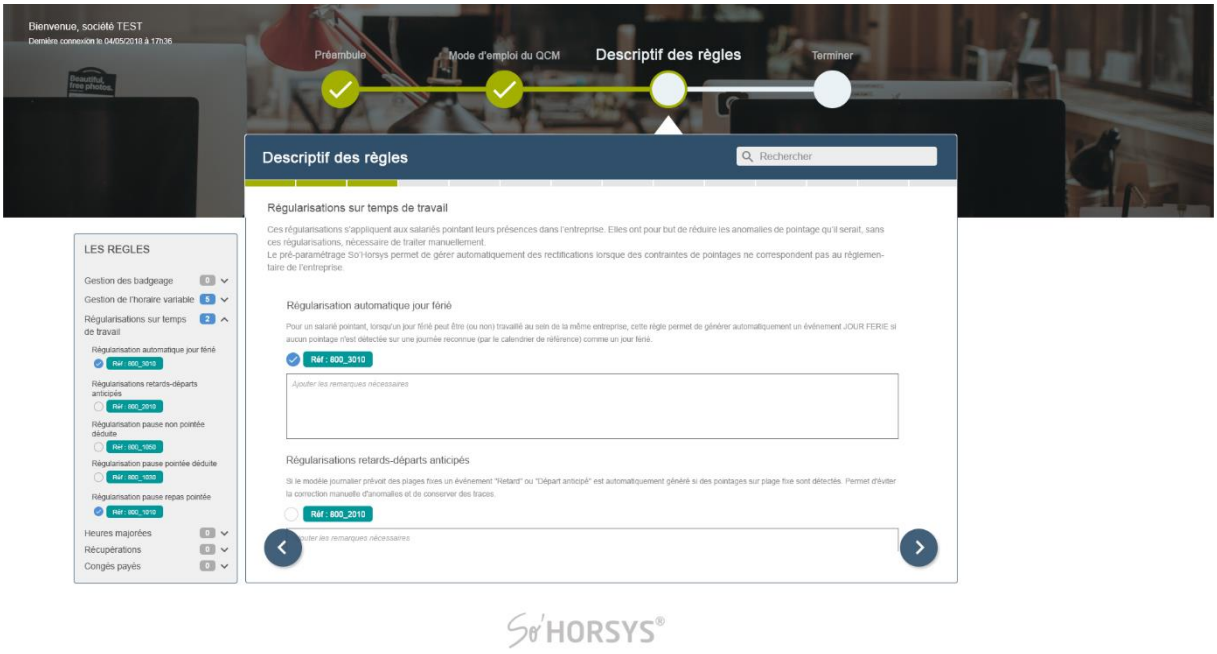
Le + pour les clients :

- Temps d'intervention plus rapide donc moins chère

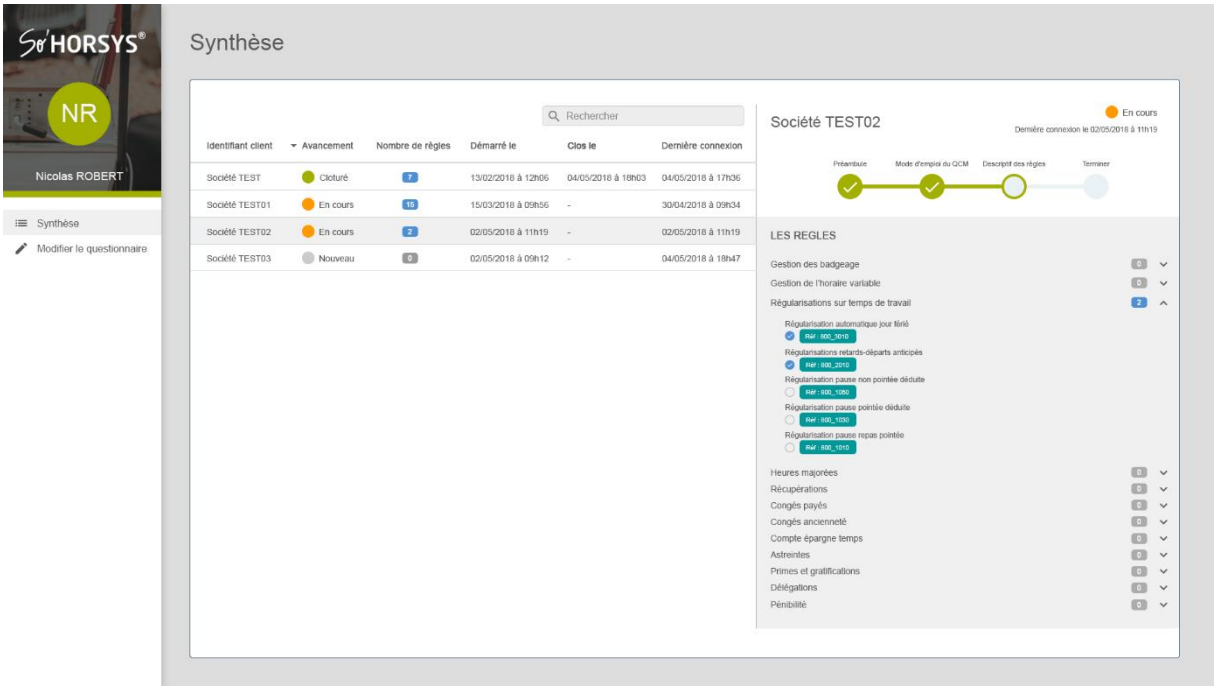
Le + pour ASYS :

- Optimisation de l'intégration
- Possibilité d'organiser le travail des intégrateurs en fonction de l'avancement des clients (gain de temps)
- Logiciel moins onéreux donc potentiel nouveaux clients

Maquette correspondant à la première interface. (Nicolas)



Maquette correspondant à la seconde interface. (Mathis)



3.SPÉCIFICATIONS

3.1 Environnement

Pour mener à bien ce projet, nous utiliserons le Framework développé en interne ainsi que son propre format de sérialisation des données.

Concernant l'éditeur, nous développerons sous Visual Studio Code.
Le projet devra répondre aux attentes sous les versions minimums de :

- Internet Explorer 11
- Firefox 48
- Chrome 50

Tous le code devra être normalisé, respecter la syntaxe utilisée au sein de l'entreprise pour permettre l'évolution ainsi que la maintenance du projet par la suite.

```
Organisation des fichiers JavaScripts


//------------------ Variables Globales ------------------

//Syntaxe d'une variable Globale
var g_UnExemple = null;

//----------------------- Objets -------------------------

function CUnExemple1(){...}

function CUnExemple2(){
    var _nEntier = 0;
    var _sChaîne = '';
    var _bBooléen = true;
    var tabExemple = new Array();
    var _oObjet = new CUnExemple3(_sChaîne);

    this.vnPublic1 = _nEntier;

    function CUnExemple3(pi_sFlux){
        ...
    }

    this.UneFonction = function(){...}
}

//------------------ Fonction Publiques ------------------

function HA6Exemple1(){...}
function HA6Exemple2(){
    g_UnExemple.UneFonction();
}
```

```
Organisation des fichiers CSS


/*-------------------- BALISES ----------------------*/
                                                    A
html                                                .|
td, p                                               .|
                                                    Z
/*---------------------- ID -------------------------*/
                                                    A
#DeuxExemple                                        .|
#UnExemple                                          .|
                                                    Z
/*-------------------- CLASSES ----------------------*/
                                                    A
.CtnProgressBar>.ProgressBar                        .|
.CtnProgressBar>.ProgressBar>.Step                  .|
                                                    Z
/*---------------- MEDIA QUERIES --------------------*/
                                                    A
@media screen and (){..}                            .|
                                                    .|
                                                    Z
```

Compétences du référentiel au cours de cette semaine :

A1.4.1
Participation à un projet

C4.1.6.1
Mettre en place et exploiter un environnement de développement

1.4.1.2
Rendre compte de son activité

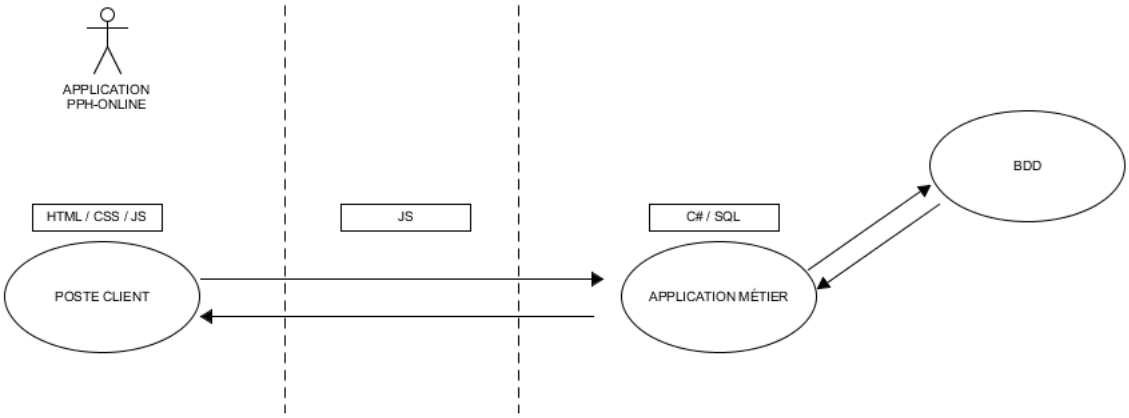
C4.1.2.2
Maquetter un élément de la solution applicative

C4.1.5.2
Développer un prototype

4. Description détaillée de l'application

Cette application devra être adaptable à tous types d'écrans, ergonomique et développée en respectant la normalisation du code de l'entreprise. De plus, tous le contenu devra être affiché de manière dynamique à partir d'un flux généré à partir d'une autre application.

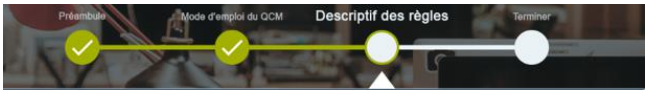
L'architecture de ce projet est la suivante :



Ce projet est réalisé dans la partie POSTE CLIENT et un flux de test est utilisé permettant de gérer le contenu dynamiquement.

Etant donné que certains éléments de chacune des interfaces sont similaires (barre de progression, gestion des règles) l'utilisation de la programmation orientée objet est utilisé en JavaScript.

Front-Office



Back-Office

Fonctionnement de l'application (Front-Office) :

Au chargement de la page, appel d'une fonction JS « HA6InitPage() » qui permet de créer et d'initialiser les objets correspondants aux éléments de la page en fonction des informations du flux d'init.

- Permettre une navigation d'étape en étape via un bouton « valider ».
- Permettre la sélection des modules souhaitées.

```
//Au chargement de la page.
function HA6initPage(){

    var sFluxPPH="1[n3$1[f$Preambule[n2$2[f$Mode d'emploi du QCM[n2$3[f$Descriptif des règles[n1$1[
    var sFluxMouchard = "Bienvenue, société TEST[f$Dernière connexion le 11/03/2019 à 17h50";

    g_oPage = new CPAGE(sFluxPPH);
    g_oMouchard = new CMOUCHARD(sFluxMouchard);

}
```

Découpage en Fonctionnalités (Front-Office) :

- Création de la structure globale de la page (conteneur principal, en-tête, pied de page)
- Création du visuel de chaque éléments (barre de progression, gestion des règles)
- Rendre la barre de progression fonctionnelle en fonction du flux
- Rendre la gestion des règles fonctionnel en fonction du flux

5. Développement des fonctionnalités :

La structure HTML de la page est plutôt basique étant donné que tout est géré dynamiquement. Voici la structure :

```
<body onload="HA6initPage()">
    <!-- En Tête -->
    <div class="header">
        <div id="Mouchard">
        </div>

        <div class="CtnProgressBar">
            <div id="ProgressBar" class="ProgressBar">
            </div>
        </div>

    </div>

    <div class="CTN-Page co_BgColorWhite">
        <!-- En-tête de la fenêtre principale -->
        <div class="TXT-EnTete co_BgDarkBlue">
            <div>Préambule</div>

        </div>

        <!-- Contenu de la fenêtre principale -->
        <article class="TXT-Contenu co_ColorGrayText">
        </article>

        <!-- Bouton validation -->
        <div id="btnValidation" class="co_BtnAction co_BtnActionTextLeft MonBoutonAddRight">
            Suivant
        </div>

    </div>

    <div id="Regles"class="Regles">
        <div class="Entete">
            LES REGLES
        </div>
    </div>

    <div class="IMG-LogoSoHorsys">
    </div>

    <div class="footer"></div>


</body>
```

Concernant les classes css, j'utilise la cascade pour cibler les éléments.
En voici un exemple correspondant au style de la structure ci-dessus :

```css
/* ----------------------------Conteneur page principal----------------------- */

.CTN-Page{...
}
.CTN-Page>.MonBoutonAddLeft{...
}

.CTN-Page>.MonBoutonAddLeft:hover{...
}

.CTN-Page>.MonBoutonAddRight{
    position: absolute;
    right: 5%;
    bottom: 5%;
    background-image: url(../../Images/Commun/check_white.svg);

    background-color: #a3b000;
}

.CTN-Page>.MonBoutonAddRight:hover{
    min-width: 60px;
}

/* Contenu Affiché dans la fenêtre principale */
.CTN-Page>.TXT-Contenu{...
}

/* Texte présent dans l'en-tête de la fenêtre principale */
.CTN-Page>.TXT-EnTete{...
}

/* ------------------------------------------------------------------------- */
```

5.1 Création de la barre de progression :

Pour mettre en place la barre de progression, j'utilise les structures HTML et CSS ci-dessous :

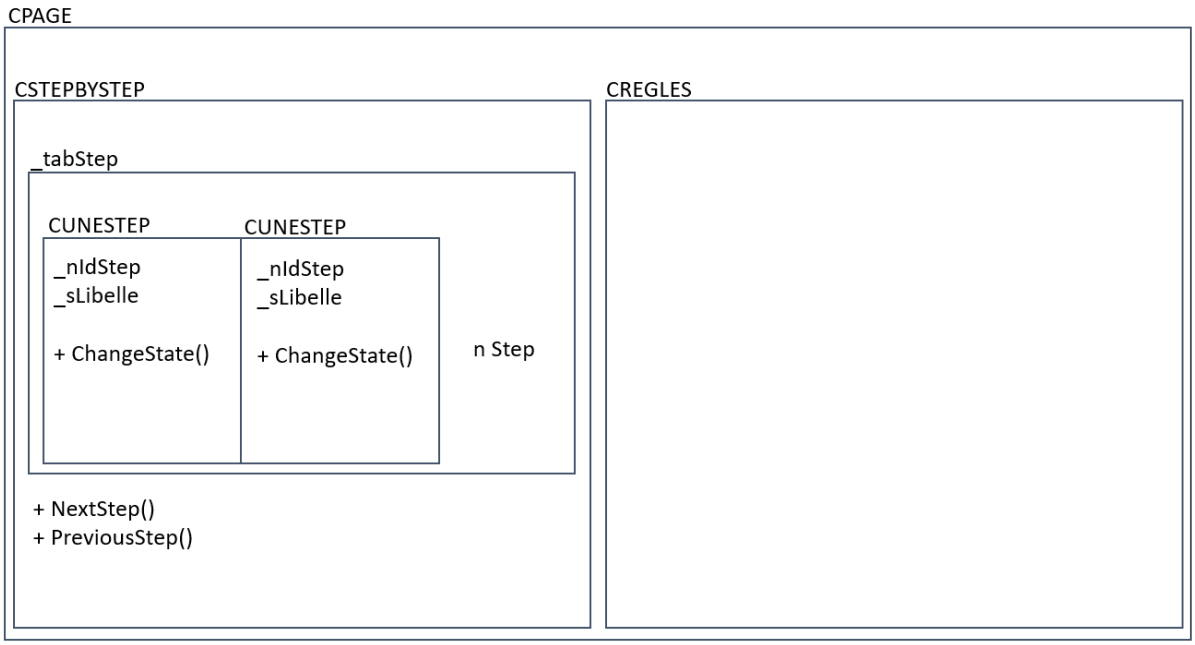
```html
<div id="ProgressBar" class="ProgressBar">
    <div class="Step" style="width: calc(25%);">
        <div class="Element">
            <div class="Circle selected"></div>
            <label class="Libelle Surbrillance">Preambule</label>
            <div class="Indicator show"></div>
        </div>
        <div class="Bar"></div>
    </div>

    <div class="Step" style="width: calc(25%);">
        <div class="Element">
            <div class="Circle"></div>
            <label class="Libelle">Mode d'emploi du QCM</label>
            <div class="Indicator"></div>
        </div>
        <div class="Bar"></div>
    </div>
</div>
```

```css
/* ----------------------------Progress-Bar----------------------------*/
#ProgressBar{...
#ProgressBar>.Step{...
#ProgressBar>.Step>.Element{...

#ProgressBar>.Step>.Element>label{...

#ProgressBar>.Step>.Element>.checkStep{...

#ProgressBar>.Step>.Element>.Circle{...

#ProgressBar>.Step>.Element>.Indicator{...

#ProgressBar>.Step>.Element>.Libelle{...

#ProgressBar>.Step>.Element>.selected{...

#ProgressBar>.Step>.Element>.Surbrillance{...

#ProgressBar>.Step>.Element>.show{...
#ProgressBar>.Step>.Bar{...

#ProgressBar .active{...
/* ------------------------------------------------------------------------*/
```

Pour le moment cette structure n'est pas générée dynamiquement, afin de créer dans un premier temps une barre ressemblant à la maquette.

Une fois la structure de la barre de progression terminée, j'ai commencé à réfléchir sur le fonctionnement de cette dernière puis à la manière dont j'allais gérer les différents états.

Pour cela j'ai donc modéliser mes futurs objets JavaScript à travers un schéma :



L'objet JavaScript correspondant à la barre de progression se nomme « CSTEPBYSTEP »

Cet objet contient un tableau d'objet « CUNESTEP » où chaque élément du tableau correspond à une étape.

L'objet « CREGLES » n'est encore pas défini.

CSTEPBYSTEP :

+ *NextStep :* Méthode permettant de passer à l'étape suivante
+ *PreviousStep :* Méthode permettant de passer à l'étape précédente

CUNESTEP :

+ *ChangeState :* Méthode indiquant le traitement à réaliser sur une étape en fonction de son état (activé, en cours, désactivé)

<u>Fonctionnement de la barre de progression (dynamique) :</u>

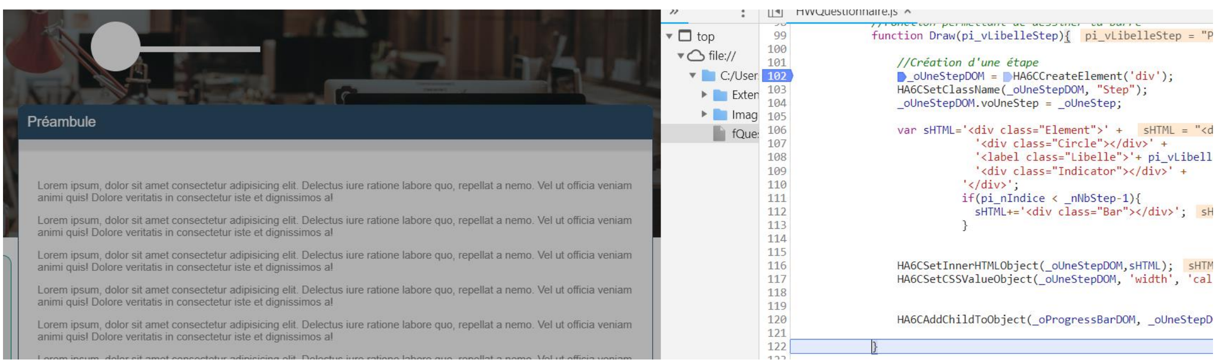
Afin de générer cette barre dynamiquement, il est nécessaire d'ajouter à la page HTML un conteneur possédant un id « ProgressBar ».
Ensuite, il faut utiliser les classes CSS correspondante (voir plus haut).

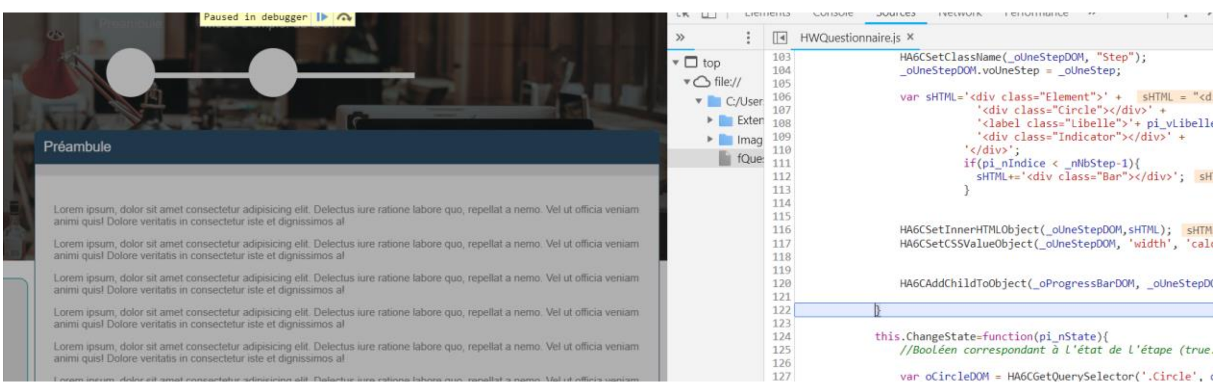
```html
<div class="CtnProgressBar">
    <div id="ProgressBar" class="ProgressBar">
    </div>
</div>
```

Puis sur la page JavaScript, il faut déclarer et instancié un nouvel objet CSTEPBYSTEP.

Pour dessiner la barre de progression, l'objet JS se base sur le nombre d'étape à afficher. Ensuite, il boucle sur chacune des étapes et appel une fonction Draw() permettant l'affichage dans le DOM d'une étape.

```javascript
function Draw(pi_vLibelleStep){

    //Création d'une étape
    _oUneStepDOM = HA6CCreateElement('div');
    HA6CSetClassName(_oUneStepDOM, "Step");
    _oUneStepDOM.voUneStep = _oUneStep;

    var sHTML='<div class="Element">' +
            '<div class="Circle"></div>' +
            '<label class="Libelle">'+ pi_vLibelleStep +'</label> ' +
            '<div class="Indicator"></div>' +
          '</div>';
          if(pi_nIndice < _nNbStep-1){
            sHTML+='<div class="Bar"></div>';
          }


    HA6CSetInnerHTMLObject(_oUneStepDOM,sHTML);
    HA6CSetCSSValueObject(_oUneStepDOM, 'width', 'calc(100% / '+ _nNbStep + ')');


    HA6CAddChildToObject(_oProgressBarDOM, _oUneStepDOM);

}
```





Compétences du référentiel au cours de cette semaine :

A1.4.1
Participation à un projet

C4.1.6.1
Mettre en place et exploiter un environnement de développement

1.4.1.2
Rendre compte de son activité

C4.1.2.2
Maquetter un élément de la solution applicative

C4.1.5.2
Développer un prototype

Pour gérer les différents états d'une étape, j'utilise trois variables constante pour :

- Une étape validée
- Une étape en cours
- Une étape non validée

Pour chacune des étapes créer via un flux spécifique, j'appelle une fonction ChangeState() qui elle permet en fonction de la valeur de la constante, applique le style correspondant à l'étape.

```
//Constantes pour gérer l'affichage
var CST_STATE = {
    OK : 1,
    EN_COURS : 2,
    NOK : 3,

    ADMIN : 1,
    CLIENT : 0
}
```

```
//Fonction pour gérer l'affichage en fonction de l'état de l'étape
this.ChangeState=function(pi_nState,pi_nMode){

    var oCircleDOM = HA6CGetQuerySelector('.Circle',_oUneStepDOM);
    var oUneBarDOM = HA6CGetQuerySelector('.Bar',_oUneStepDOM);

    //Application des styles
    switch(pi_nState){}
}
```

```
//Appel de la fonction ChangeState() pour chacune des étapes
for(var i=0; i< tabStep.length ;i++){
    var nIndexStep = i+1;
    var oUneStepDOM = tabStep[i];

    if(nIndexStep < pi_nStepEnCours){
        //Etape Validée
        oUneStepDOM.voUneStep.ChangeState(CST_STATE.OK,pi_nMode);
    }else if(nIndexStep == pi_nStepEnCours){
        //Etape en cours
        oUneStepDOM.voUneStep.ChangeState(CST_STATE.EN_COURS,pi_nMode);
    }else if(nIndexStep > pi_nStepEnCours){
        //Etape non validée
        oUneStepDOM.voUneStep.ChangeState(CST_STATE.NOK,pi_nMode);
    }

}
```

Cet objet possède aussi deux méthodes permettant la navigation entre chacune des étapes :

- NextStep()
- PreviousStep()

FLUX ETAPE EN COURS		
Objets	*Etape en cours*	
	Fields	
	1	EtapeEnCours

Ainsi qu'un mode client et un mode administrateur (suppression de l'indicateur ainsi que la surbrillance du libelle) pour permettre à la partie backoffice de réutiliser ce même objet.

```
this.NextStep=function(){

    _nStepEnCours++;
    if(_nStepEnCours>_nNbStep){
        _nStepEnCours = _nNbStep;
    }
    SetStepEnCours(_nStepEnCours);
}

this.PreviousStep=function(){

    _nStepEnCours--;
    if(_nStepEnCours<1){
        _nStepEnCours = 1;
    }
    SetStepEnCours(_nStepEnCours);
}
```

Pour le bon fonctionnement de cet objet, deux flux sont requis :

- Le premier pour initialiser toutes les étapes
- Le second pour savoir quelle est l'étape en cours

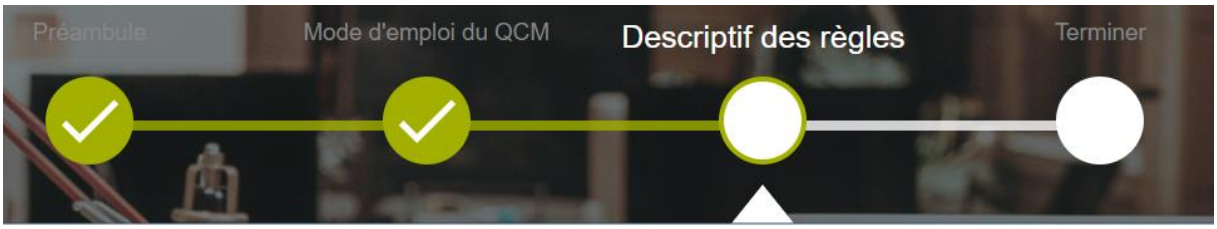
En voici les schémas :

FLUX INIT ETAPES		
Objet 1 à n	*Une Etape*	
	Fields	
	1	Libelle
	2	Type Etape

Flux réel :

```
//Flux Progress Bar
sFluxInitPGB = "Préambule[f$0[o$Mode d'emploi du QCM[f$0[o$Descriptif des règles[f$1[o$Terminer[f$0[o$";
sFluxEtapeEnCours = "3";
```

Résultats final :



Création d'un nouvel objet correspondant aux règles.
J'ai décidé de créer un seul objet pour gérer le menu des règles ainsi que la page principale contenant tous les modules concernant une règle.

Seul la création de l'objet changera en fonction de son utilisation.
Un mode administrateur et client seront aussi présent et seront déterminer à travers la création de l'objet pour permettre à la partie backoffice de réutiliser cet objet.

Dans un premier temps, j'ai réfléchi à la structure de mon objet et à son fonctionnement avec le flux que nous avons créé au préalable avec Mathis pour ainsi utiliser un flux commun.
Le flux sera différent en fonction du type d'étape, actuellement nous possédons deux types d'étapes :

- Type Texte (Objet crée par Mathis)
- Type Règle (Objet crée par Nicolas)

Voici le Schéma du flux :

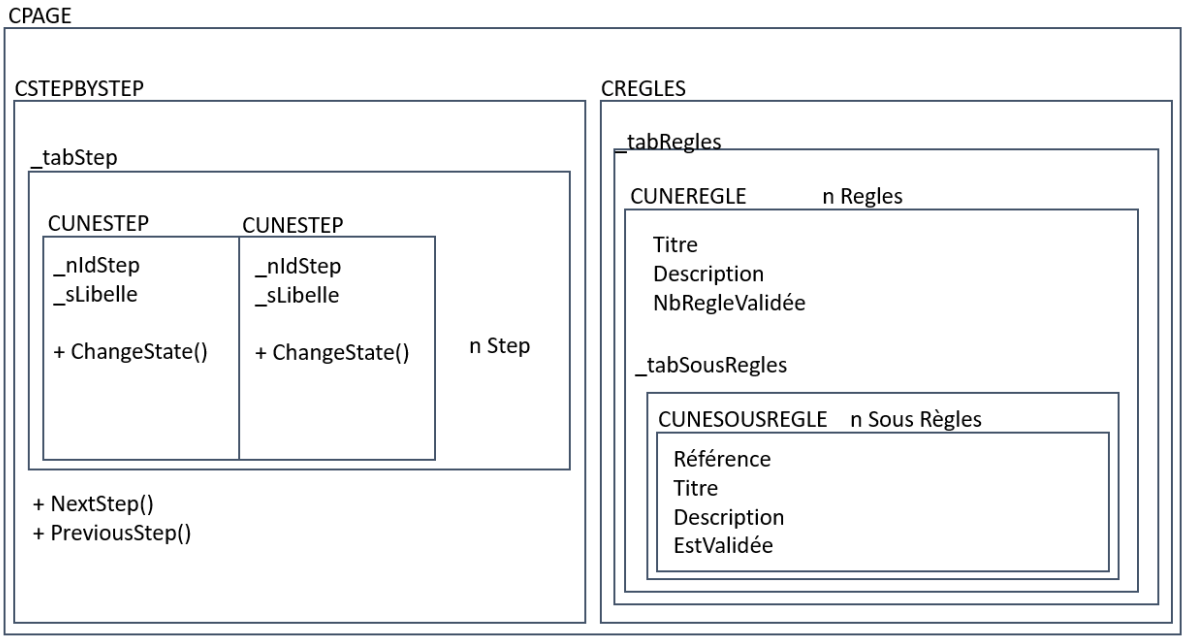
Flux Etape de type Texte :

```
sFluxInitEtape = "0[a$Préambule[f$Lorem ipsum dolor sit amet consectetur";
```

```
FluxInitEtape = "1[a$Régularisation sur le temps de travail[f$Ces regularisations s'appliquent …[f$2[p$800_2010[f$Régularisation retard-départs anticipé[f$Si le modèle ...[f$
                "[f$Test Commentaire dynamique[e$500_2010[f$Régularisation retard-départs anticipé[f$Si le modèle ...[f$1[f$Test Commentaire dynamique";
```

Flux Etape de type règles :

Schéma de la structure de mon objet CREGLES

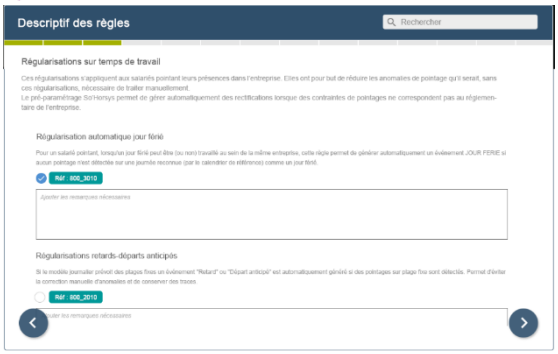


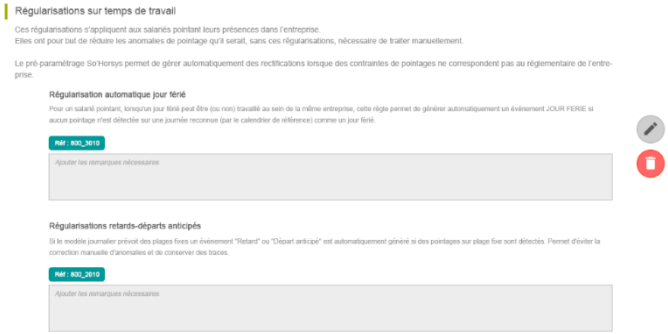
Comme pour la barre de progression, différent affichages un peu plus complexes sont à gérer :

- Le menu des règles détaillé pour le client
- Le menu des règles détaillé pour l'administrateur
- Le menu des règles simple pour l'administrateur
- La page principale client
- La page principale administrateur

5.2 Création du menu des règles :

```
<div id="Regles"> == $0
  <div class="Entete">
            LES REGLES
          </div>
<div class="Element">
  <div id="CtnElement0" class="CtnElement">
     <div class="FamilleRegles">Régularisation sur le temps de travail</div>
    <div class="CtnRight">
       <div id="NbRegle0" class="NbRegles" style="background-color: rgb(79, 144, 211);">2</div>
       <div id="tool0" class="ToolDrop"></div>
     </div>
   </div>
  <div class="UneRegle">
     <div class="Libelle">Régularisation retard-départs anticipé</div>
     <div id="chk800_2010" class="Check Checked"></div>
     <div class="Reference">Réf :800_2010</div>
   </div>
  <div class="UneRegle">…</div>
  </div>
<div class="Element">…</div>
<div class="Element">…</div>
<div class="Element">…</div>
<div class="Element">…</div>
</div>
```





Voici la structure HTML que je dois générer dynamiquement à partir des éléments de mon flux :

```
//Constante paramétrage affichage
var CST_AFFICHAGE = {
    CLIENT : 0,
    ADMIN : 1,

    MENU_SIMPLE : 10,
    MENU_DETAIL : 20,
    MAIN_PAGE : 30
}
```

Compétences du référentiel au cours de cette semaine :

A1.4.1
Participation à un projet

C4.1.6.1
Mettre en place et exploiter un environnement de développement

1.4.1.2
Rendre compte de son activité

C4.1.2.2
Maquetter un élément de la solution applicative

C4.1.5.2
Développer un prototype

Afin de générer cette structure dynamiquement à partir des éléments du flux, je crée plusieurs fonctions nommées Draw() qui servirons à dessiner le menu des règles.

```
//Dessine l'objet CUNEREGLE
function Draw(pi_sLibelle,pi_sDescription,pi_nNbRegle){

    switch (_nMode) {
        //Si mode Client
        case CST_AFFICHAGE.CLIENT:

            switch (_nAffichage){ ···
            }

            break;
        //Si mode administrateur
        case CST_AFFICHAGE.ADMIN:
            switch (_nAffichage){ ···
            }

            break;
    }

}
```

```
//Retourne une chaîne correspondant à la structure HTML du menu simplifié
function getAffichageMenuSimple(pi_sLibelle){
    var sHTML =     '<div id="CtnElement' + pi_nIndice + '" class="CtnElement">'  +
                        '<div class="FamilleRegles">'       +
                            pi_sLibelle                     +
                        '</div>'                            +
                    '</div>';
    return sHTML;
}
//Retourne une chaîne correspondant à la structure HTML du menu détaillé
function getAffichageMenuDetail(pi_sLibelle,pi_nNbRegle){
    var sHTML = '<div id="CtnElement' + pi_nIndice +'" class="CtnElement">'     +
                    '<div class="FamilleRegles">'                               +
                        pi_sLibelle                                             +
                    '</div>'                                                    +
                    '<div class="CtnRight">'                                    +
                    '<div id="NbRegle' + pi_nIndice + '" class="NbRegles">'     +
                        pi_nNbRegle                                             +
                    '</div>'                                                    +
                '<div id="tool' + pi_nIndice + '" class="ToolDrop"></div>'      +
                '</div>'                                                        +
                '</div>';



    return sHTML;
}
```

Dans un premier temps, il faut dessiner les règles :

Cette fonction Draw() appartient à l'objet CUNEREGLE et permet de crée la structure suivante (mode détaillé) :

Puis, une fois les règles créée je réalise un traitement similaire mais cette fois ci avec les références appartenant aux règles.
En utilisant toujours une fonction Draw(), je boucle sur les sous règles présentes dans mon flux pour créer de nouveaux objet CUNESOUSREGLE :

```
//Pour chacune des sous-règle, on crée un objet CUNESOUSREGLE
var tabElement = HA6WF_ELEMENT(tabParent[1]);
for (var i = 0; i < tabElement.length; i++) {

    _tabSousRegles.push(new CUNESOUSREGLE(tabElement[i],_oUnElementDOM));

}
```

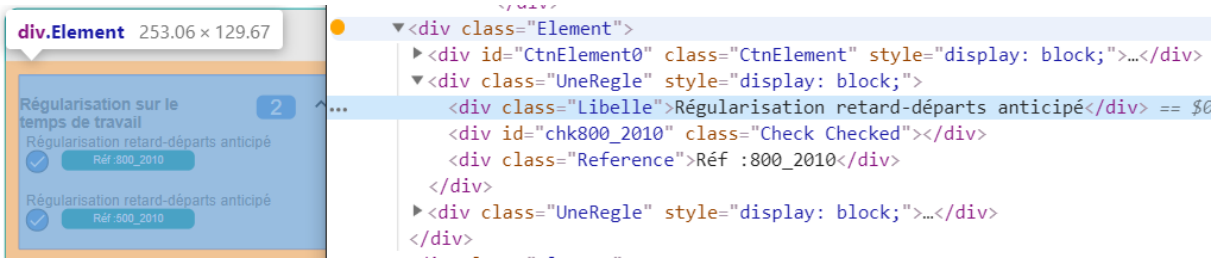
```
//Dessine une SousRegle
function Draw(pi_sReference,pi_sLibelle,pi_sDescription,pi_bEstCheck,pi_sCommentaire){
    switch (_nMode) {
        case CST_AFFICHAGE.CLIENT:
            switch (_nAffichage){
                case CST_AFFICHAGE.MENU_DETAIL:
                    var oUneSousRegleDOM = HA6CCreateElement('div');
                    HA6CSetClassName(oUneSousRegleDOM,'UneRegle');
                    var sHTML = getAffichageMenuSousReglesSimple(pi_sLibelle,pi_sReference);
                    HA6CSetInnerHTMLObject(oUneSousRegleDOM,sHTML);
                    HA6CAddChildToObject(_oUnElementDOM,oUneSousRegleDOM);
                break;
```

```
//Affiche sous règles
function getAffichageMenuSousReglesSimple(pi_sLibelle,pi_sReference){
    var sHTML =     '<div class="Libelle">'+
                    pi_sLibelle +
                    '</div>'+
                    '<div id="chk' + pi_sReference +'" class="Check"></div>'+
                    '<div class="Reference">Réf :'+
                    pi_sReference +
                    '</div>'+
                '</div>';
    return sHTML;
}

//Affiche sous règles avec une zone permettant de visualiser les commentaires (si il y en a)
function getAffichageMenuSousReglesDetail(pi_sLibelle,pi_sReference,pi_sCommentaire){
    var sHTML =     '<div class="Libelle">'+
                    pi_sLibelle +
                    '</div>'+
                    '<div id="chk' + pi_sReference +'" class="Check"></div>'+
                    '<div class="Reference">Réf :'+
                    pi_sReference +
                    '</div>';
        if(!HA6CIsUndefined(pi_sCommentaire)){
            sHTML +='<div class="Commentaire">' + pi_sCommentaire + '</div>'+
                    '</div>';
        }

    return sHTML;
}
```

Puis la fonction Draw() de cet objet s'occupe de dessiner la structure :

Cette fonction gère aussi le fait qu'un checkbox est sélectionné ou non.

Voici la structure finale :



5.3 Création de la page principale :

La page principale fonctionne exactement sur le même principe que le menu des règles, simplement la structure HTML est différente ainsi que l'appel de l'objet CREGLES
Signature de l'objet CREGLES :

Voici donc l'appel de cet objet afin d'afficher la page principale en tant que client (ma partie)

```
var _opageprinc = new CREGLES(HA6CGetId('Main_Page'),sFluxMainPage,30,0);
```

```
/*OBJET CREGLES :

pi_ObjectIHM : Objet DOM contenant l'objet CREGLES
pi_sFlux : Flux
pi_nAffichage : Entier correspondant au type affichage
                10=>Menu Simplifié
                20=>Menu Detaillé
                30=>Page Principale
pi_nMode : Type d'utilisateur
                0=>Client
                1=>Administrateur
*/
function CREGLES(pi_ObjectIHM,pi_sFlux,pi_nAffichage,pi_nMode){ ···
}
```

Dans un premier temps, il faut afficher la règle en cours, pour cela la fonction Draw() de l'objet CUNEREGLE dessinera cette structure :

```
<!-- Contenu de la fenêtre principale -->
<div id="Main_Page" class="TXT-Contenu co_ColorGrayText"> == $0
  ▼<div class="Ctn-Regle">
      <div class="TxtRegle">Régularisation sur le temps de travail</div>
    ▶<div class="Txt-Description">…</div>
    </div>
```

```
//Et affichage du contenu principal
case CST_AFFICHAGE.MAIN_PAGE:
    var oCtnRegleDOM = HA6CCreateElement('div');
    HA6CSetClassName(oCtnRegleDOM,'Ctn-Regle');

    var sHTML = getAffichageMainPageRegle(pi_sLibelle,pi_sDescription,pi_nNbRegle);

    HA6CSetInnerHTMLObject(oCtnRegleDOM,sHTML);
    HA6CAddChildToObject(_oCtnDOM, oCtnRegleDOM);

    var oCtnSousReglesDOM = HA6CCreateElement('div');
    HA6CSetClassName(oCtnSousReglesDOM,'Ctn-SousRegles');
    HA6CAddChildToObject(_oCtnDOM, oCtnSousReglesDOM);
break;
```

```
//Retourne une chaîne correspondant à la structure HTML du contenu principal
function getAffichageMainPageRegle(pi_sLibelle,pi_sDescription){

    sHTML=  '<div class="TxtRegle">' + pi_sLibelle + '</div>' +
            '<div class="Txt-Description">' + pi_sDescription + '</div>';

    return sHTML;

}
```

Puis, il nous faut afficher chacune des références appartenant à la règle en cours, avec la possibilité de mettre une remarque.


```
▼<div class="Ctn-SousRegles">
  ▼<div class="Ctn-UneSousRegle"> == $0
      <div class="Txt-SousRegle">Régularisation auto jour férié</div>
    ▶<div class="Txt-Description">…</div>
      <div id="Mainchk800_3010" class="Check Checked"></div>
      <div class="Txt-Reference">800_3010</div>
      <textarea name="TextBox-800_2010" class="Txt-Commentaire" placeholder="Ajouter les remarques nécessaire"></textarea>
    </div>
  ▶<div class="Ctn-UneSousRegle">…</div>
  </div>
```

Voici la structure HTML :

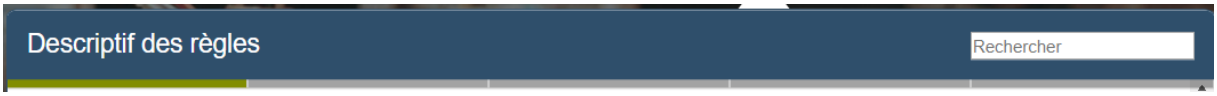
Pour générer cette structure, je fais appel à la fonction Draw () de l'objet CUNESOUSREGLE qui se charge de dessiner chacune des références.

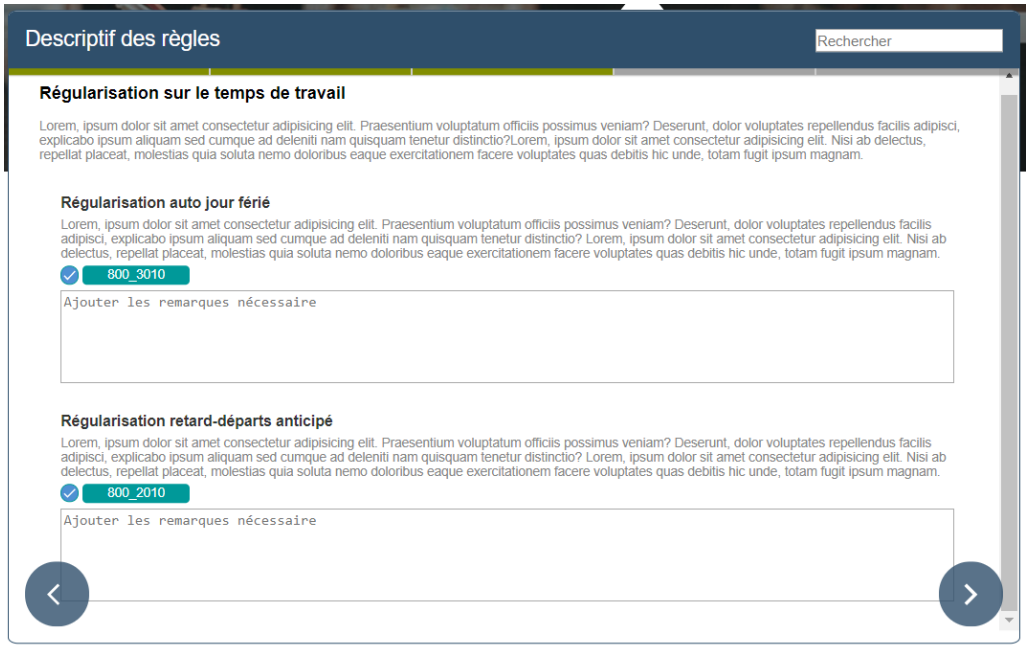
```
case CST_AFFICHAGE.MAIN_PAGE:

    var oCtnUneSousRegleDOM = HA6CCreateElement('div');
    HA6CSetClassName(oCtnUneSousRegleDOM,'Ctn-UneSousRegle');

    var sHTML = getAffichageMainPageSousRegleClient(pi_sLibelle,pi_sReference,pi_sDescription);

    HA6CSetInnerHTMLObject(oCtnUneSousRegleDOM,sHTML);
    HA6CAddChildToObject(HA6CGetQuerySelector('.Ctn-SousRegles',_oCtnDOM),oCtnUneSousRegleDOM);

break;
```

```
//Retourne la structure HTML de la page principale
function getAffichageMainPageSousRegleClient(pi_sLibelle,pi_sReference,pi_sDescription,pi_sCommentaire){
    sHTML = '<div class="Txt-SousRegle">' + pi_sLibelle + '</div>' +
            '<div class="Txt-Description">' + pi_sDescription + '</div>'      +
            '<div id="Mainchk' + pi_sReference +'" class="Check"></div>'          +
            '<div class="Txt-Reference">' + pi_sReference + '</div>'+
            '<textarea name="TextBox-800_2010" class="Txt-Commentaire" placeholder="Ajouter les remarques nécessaire"></textarea>';

    return sHTML;
}
```

5.4 Création de l'objet CPROGRESSREGLE :

Cet objet correspond à l'état d'avancement de la consultation des règles.
La navigation se fait soit avec les boutons de navigations, soit en sélectionnant la règle souhaitée dans le menu.





```javascript
//Objet correspondant à l'état d'avancement des règles
function CPROGRESSREGLE(pi_nRegles,pi_nIndexRegleEnCours,pi_oParentObject){
    _oProgressRegleDOM = null;
    _nNombreDeRegle = pi_nRegles;
    _nIndexRegleEnCours = pi_nIndexRegleEnCours;
    _tabRegles = new Array();

    Draw();
    _oProgressBarDOM = HA6CGetQuerySelector('Ctn-ProgressRegles');
    _tabRegles = HA6CGetQuerySelectorAll('.item',_oProgressBarDOM);

    function Draw(){ ···
    }

    function SetActive(pi_oObject,pi_nIndex){ ···
    }


    this.Next = function(){ ···
    }

    this.Previous = function(){ ···
    }

    this.GetRegles = function(){ ···
    }


}
```

Voici la structure de mon objet JavaScript :

```javascript
//Gestion de l'état de la règle
function SetActive(pi_oObject,pi_nIndex){

    if(pi_nIndex == 0){
        HA6CSetClassAdd(pi_oObject,'active');
    }
    else if(pi_nIndex < _nIndexRegleEnCours){
        HA6CSetClassRemove(pi_oObject,'inactive');
        HA6CSetClassAdd(pi_oObject,'active');
        }
        else{
            HA6CSetClassAdd(pi_oObject,'inactive');
            HA6CSetClassRemove(pi_oObject,'active');
        }
}

//Passe à la prochaine règle
this.Next = function(){

    if(_nIndexRegleEnCours == _tabRegles.length){
        _nIndexRegleEnCours = _nIndexRegleEnCours;
    }else{
        _nIndexRegleEnCours++;
        var nIndexNextRegle = _nIndexRegleEnCours-1;
        SetActive(_tabRegles[nIndexNextRegle],nIndexNextRegle);
    }
}
//Revient à la règle précédente
this.Previous = function(){

    if(_nIndexRegleEnCours == 1){
        _nIndexRegleEnCours = 1;
    }else{
        _nIndexRegleEnCours--;
        SetActive(_tabRegles[_nIndexRegleEnCours],_nIndexRegleEnCours);
    }
}
```

Cet objet est dynamique et s'adapte au nombre de règles présentes dans le flux d'initialisation.

5.5 Affichage du Menu des règles sur mobile :

L'affichage du menu des règles sur mobile dans l'état n'est pas possible en raison de la taille d'écran disponible. J'utilise donc un objet CCAROUSEL développé par ASYS qui permet la création d'un carrousel avec comme paramètre d'entrée un objet DOM, un tableau d'objet élément contenant un code, ainsi qu'un libellé, et enfin le code par défaut à afficher.

Voici sa signature :

```javascript
// Objet Carousel
// pi_oObjectIHM = objet IHM
// pi_tabElements = tableau d'objets de type
//      objet ELEMENT{
//          vsCode = Code de l'élément
//          vsLibelle = Libelle de l'élément
//          + autres si besoin car ne sera pas utilisé ici
//}
// pi_sCodeDefault = code par défaut à appliquer
function CCAROUSEL(pi_oObjectIHM, pi_tabElements, pi_sCodeDefault){···
}
```

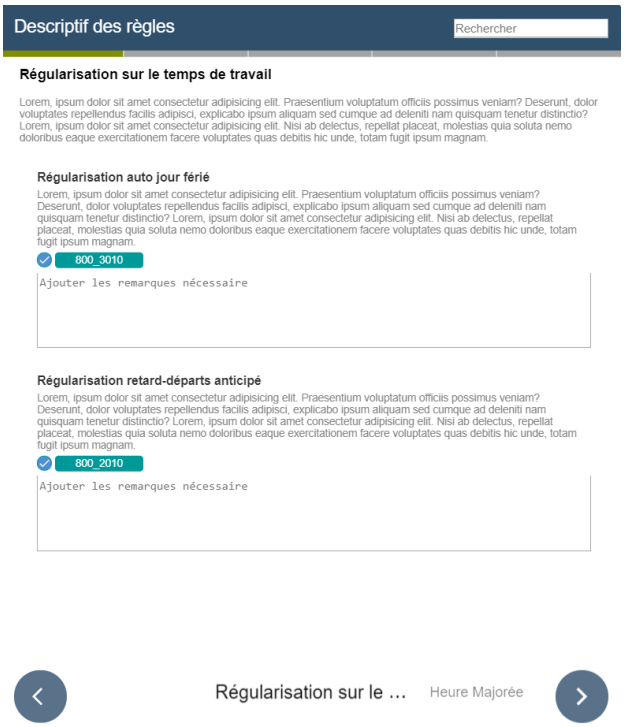
Initialisation d'un tableau d'objet CUN_ELT_CAROUSEL

```javascript
function CUN_ELT_CAROUSEL(pi_sCode, pi_sLibelle){
        this.vsCode = pi_sCode;             // Code de l'élément
        this.vsLibelle = pi_sLibelle;       // Libellé de l'élément
    }
```

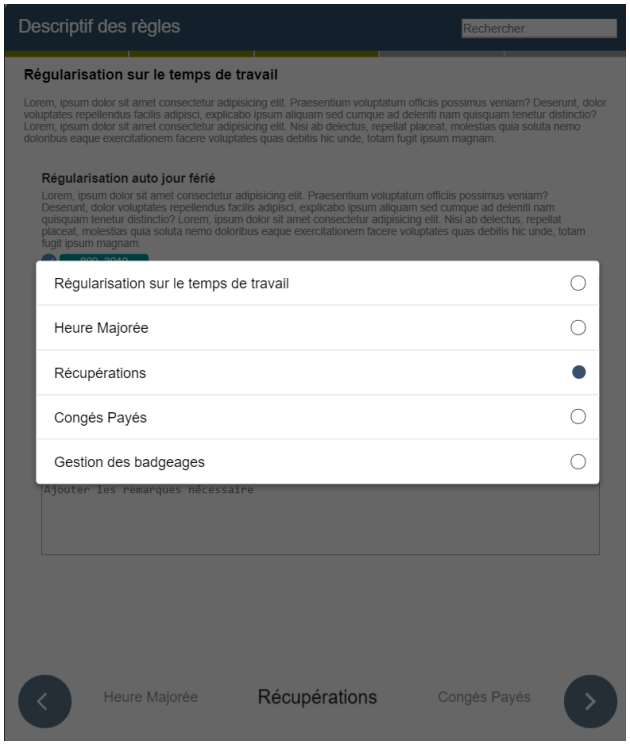
```
var tabElementCaroussel = new Array();
for (var i = 0; i < _oMenuRegles.GetRegles().length; i++) {
    tabElementCaroussel.push(new CUN_ELT_CAROUSEL(i,_oMenuRegles.GetRegles()[i].vsLibelle));
}
```

```
_oCarroussel = new CCAROUSEL(HA6CGetId('Carousel'),tabElementCaroussel,"0");
```

Instanciation de l'objet CCAROUSEL :



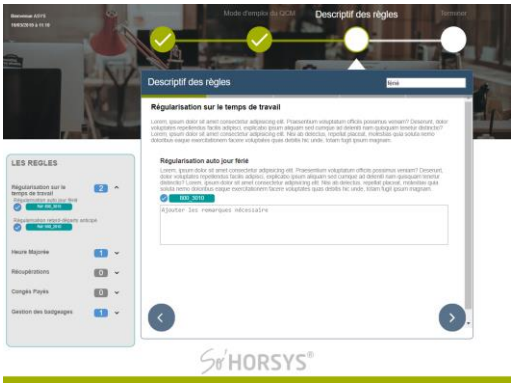
Et voici le



résultat final sur un affichage mobile :

Sur le clic d'une règle une fenêtre pop-up apparaît et permet ainsi la sélection d'une règle.

5.6 Création de la barre de recherche :



La barre de recherche est générée uniquement sur une étape de type règle, j'ai donc crée un objet spécifique à la barre de recherche :

```
function CSEARCH(){
    var _oSearchDOM = null;
    Draw()
    if(!HA6CIsUndefined(HA6CGetId('SearchTxt'))){···
    }

    //Dessine la barre de recherche
    function Draw(){

        var _oCtnSearchDOM = HA6CCreateElement('div');
        HA6CSetClassName(_oCtnSearchDOM,'SearchZone');
        HA6CSetInnerHTMLObject(_oCtnSearchDOM,GetSearch());
        HA6CAddChildToObject(_oEnTeteEtapeDOM,_oCtnSearchDOM);

    }

    function GetSearch (){
        var sHTML = '<input id="SearchTxt" type="text" placeholder="Rechercher" class="SearchZoneTxt">';
        return sHTML;
    }


    _oSearchDOM.onkeyup=function(){···
   }
}
```

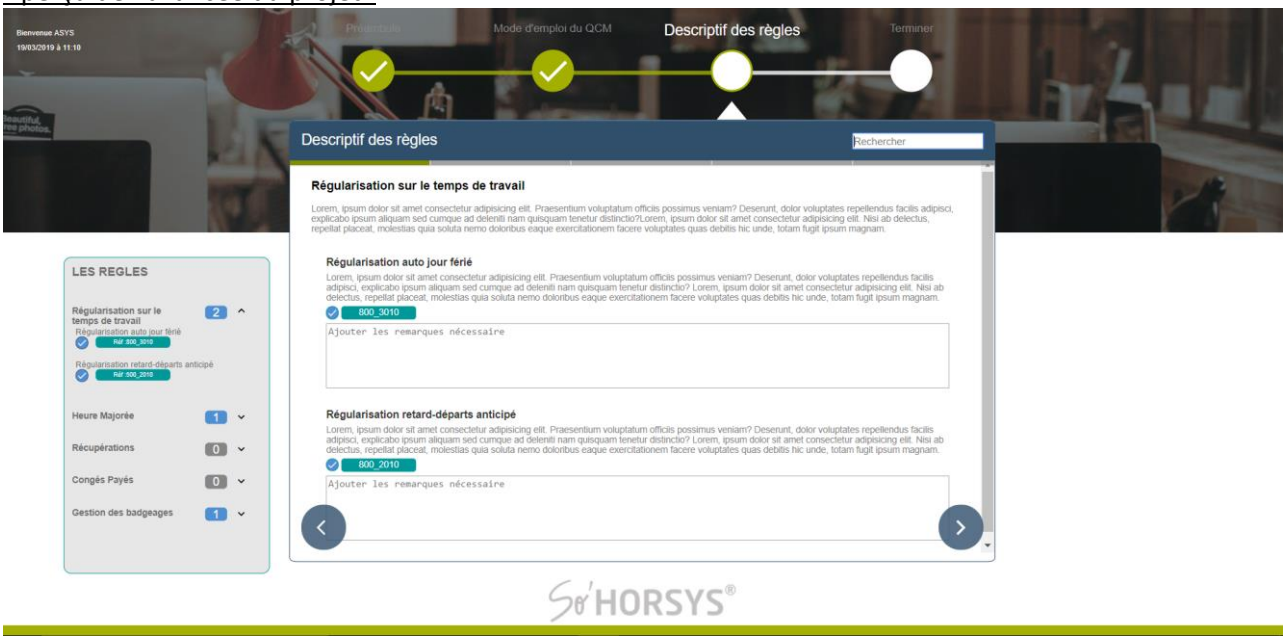
La fonction de recherche est déjà présente dans le framework de l'entreprise, j'ai simplement besoin de l'appeler au moment de la saisie.

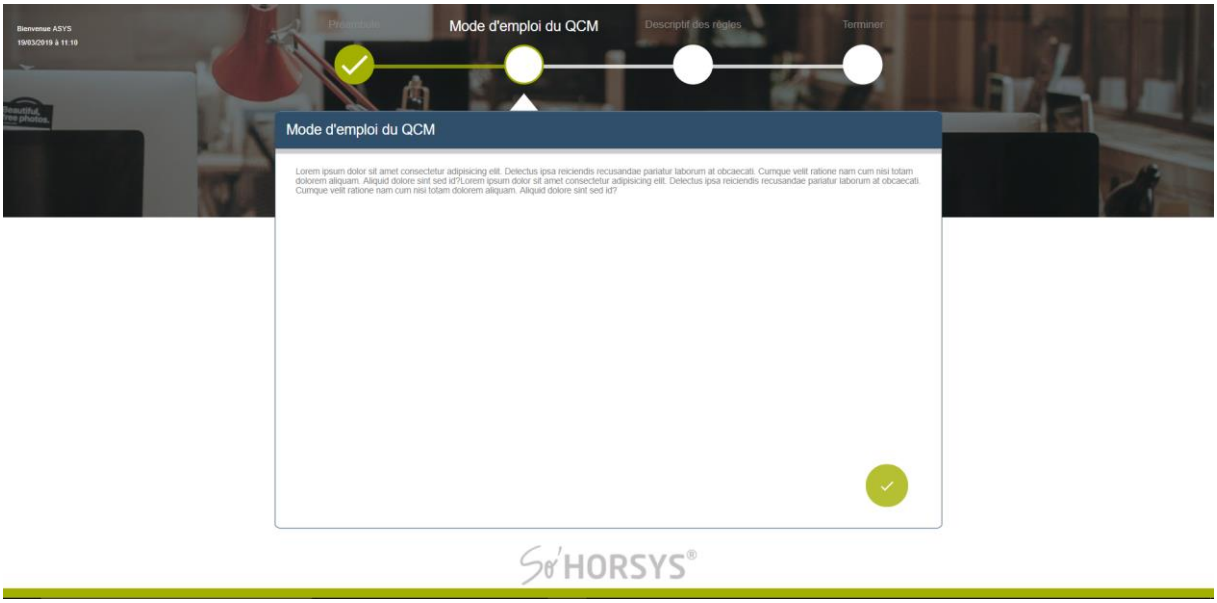
```
_oSearchDOM.onkeyup=function(){
    OnSearch(this.value);
}
```

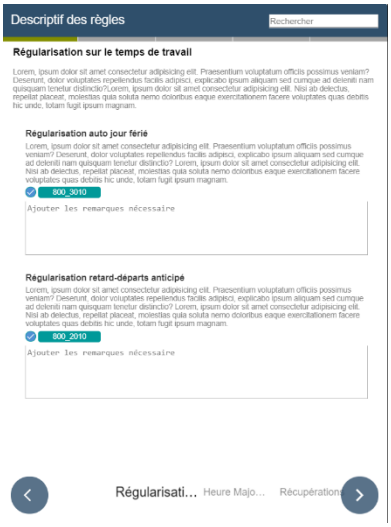
```
function OnSearch(pi_sSearchtext){
    HA6COnSearchListeDiv(pi_sSearchtext,'Ctn-UneSousRegle');
}
```

```
/* Pour une recherche texte dans une liste de Div ayant le même nom de class
function HA6COnSearchListeDiv(pi_sStringRecherche, pi_sNomClass) {···
}
```

Aperçu de l’avancée du projet :






Descriptif des règles
Rechercher
Régularisation sur le temps de travail
Lorem, ipsum dolor sit amet consectetur adipisicing elit. Praesentium voluptatum officiis possimus veniam? Deserunt, dolor voluptates repellendus facilis adipisci, explicabo ipsum aliquam sed cumque ad deleniti nam quisquam tenetur distinctio?Lorem, ipsum dolor sit amet consectetur adipisicing elit. Nisi ab delectus, repellat placeat, molestias quia soluta nemo doloribus eaque exercitationem facere voluptates quas debitis hic unde, totam fugit ipsum magnam.
Régularisation auto jour férié
Lorem, ipsum dolor sit amet consectetur adipisicing elit. Praesentium voluptatum officiis possimus veniam? Deserunt, dolor voluptates repellendus facilis adipisci, explicabo ipsum aliquam sed cumque ad deleniti nam quisquam tenetur distinctio? Lorem, ipsum dolor sit amet consectetur adipisicing elit. Nisi ab delectus, repellat placeat, molestias quia soluta nemo doloribus eaque exercitationem facere voluptates quas debitis hic unde, totam fugit ipsum magnam.
800_3010
Ajouter les remarques nécessaire
Régularisation retard-départs anticipé
Lorem, ipsum dolor sit amet consectetur adipisicing elit. Praesentium voluptatum officiis possimus veniam? Deserunt, dolor voluptates repellendus facilis adipisci, explicabo ipsum aliquam sed cumque ad deleniti nam quisquam tenetur distinctio? Lorem, ipsum dolor sit amet consectetur adipisicing elit. Nisi ab delectus, repellat placeat, molestias quia soluta nemo doloribus eaque exercitationem facere voluptates quas debitis hic unde, totam fugit ipsum magnam.
800_2010
Ajouter les remarques nécessaire
Régularisati...
Heure Majo...
Récupération


Mode d'emploi du QCM
Lorem ipsum dolor sit amet consectetur adipisicing elit. Delectus ipsa reiciendis recusandae pariatur laborum at obcaecati. Cumque velit ratione nam cum nisi totam dolorem aliquam. Aliquid dolore sint sed id?Lorem ipsum dolor sit amet consectetur adipisicing elit. Delectus ipsa reiciendis recusandae pariatur laborum at obcaecati. Cumque velit ratione nam cum nisi totam dolorem aliquam. Aliquid dolore sint sed id?

Compétences du référentiel au cours de cette semaine :

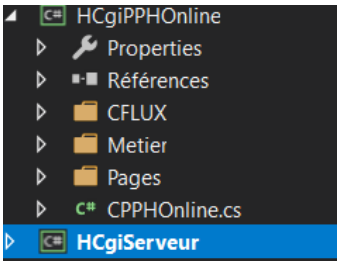
A1.4.1
Participation à un projet
A4.2.1
Analyse et correction d'un dysfonctionnement, d'un problème de qualité
C4.1.6.1
Mettre en place et exploiter un environnement de développement
1.4.1.2
Rendre compte de son activité
C4.1.2.2
Maquetter un élément de la solution applicative
C4.1.5.2
Développer un prototype
C4.1.6.2
Mettre en place et exploiter un environnement de test

Partie Application Métier :

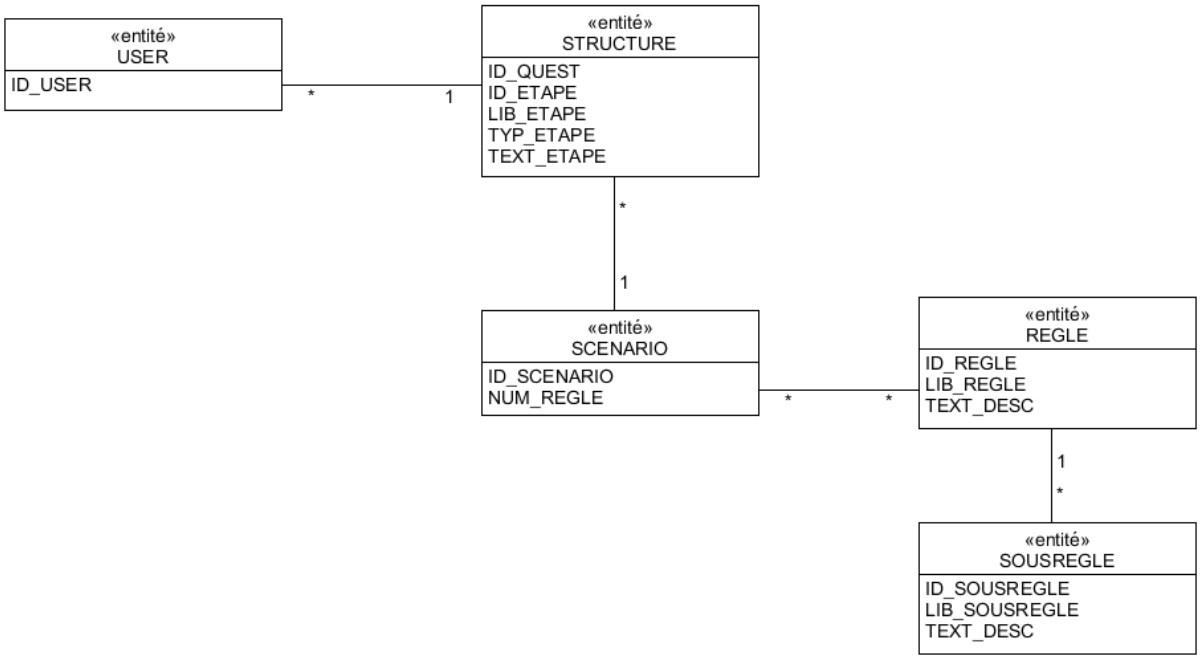
Dans cette partie, j'ai principalement abordé le raisonnement métier de l'application à destination de l'UI.

En premier lieu, mon responsable m'a fait plusieurs démonstrations du fonctionnement de l'application métier développée en C#. Cette application possède une structure proche du modèle MVC (Model View Controller), elle se décompose en 3 couches :

- FLUX (équivalent du Controller) : Gestion de la donnée, sérialisation des flux
- METIER (équivalent du Model) : Accès Base de données
- PAGES (équivalent du View) : Gestion des pages



La partie UI étant terminée, j'ai commencé à réfléchir à la structure de la future base de données. Voici une structure (temporaire) que j'ai réalisé :



La table USER est quasiment vide car elle est déjà créée par la société mais je n'ai encore pas connaissance de sa structure.

Cette structure me permettra dans un premier temps de générer mes flux dynamiquement via l'application métier (en C#) et par la suite de réalisé tous les traitements nécessaires.

Voici un exemple d'appel métier pour obtenir le flux correspondant à ma barre de progression :

Fonction exécuté au chargement de la page (fichier JS)
La fonction HFonction permet un appel métier de la fonction ''OnInitialise''. En cas de succès, un appel à la fonction ''OnInitialiseDone'' est effectué.

```
function OnInitialise(){
    var oParam = new HA6WF_CParametresUI();
    var HFonction = function(){ HWOpen("OnInitialise",OnInitialiseDone,oParam);};
    HFonction();
}
```

Dans l'application métier, la fonction OnInitialise de la page fQuestionnaire et appelée :
Cette fonction ensuite appelle la méthode OnInitialise de l'objet oFlux

```
public string OnInitialise()
{
    return oFlux.OnInitialise();
}
```

```
public string OnInitialise()
{

    CQuestionnaire oQuest = new CQuestionnaire(oMod_PPHOnline);
    _lstEtapes = oQuest.GetEtapes(1);

    _AddParamRetour(GetFluxEtapes());   //Obtient le flux pour la barre de progression
    _AddParamRetour(GetEtapeEnCours()); //Obtient l'ID de l'étape en cours
    _AddParamRetour(GetMainPage());     //Obtient le flux correspondant au contenu de la page principale
    _AddParamRetour(GetInitEtape(_lstEtapes[0])); //Initialise l'étape en cours

    return _SerialiseDialogRetourSelonMessageOK();

}
```

L'objet oQuest correspond à la classe CQuestionnaire de la couche métier (accès bdd).
La liste des étapes est donc affectée via une une méthode ''GetEtapes '' de la classe CQuestionnaire.

```
public List<CUneEtape> GetEtapes(short pi_nIdQuestion)
{

    List<CUneEtape> lstEtapes = new List<CUneEtape>();

    try
    {
        string sRequete = "SELECT ID_ETAPE, LIB_ETAPE, TYP_ETAPE,TEXT_ETAPE, FK_SCENARIO FROM STRUCTURE WHERE ID_QUEST =" + pi_nIdQuestion;
        oModule.voBase.Select(sRequete);
        while (oModule.voBase.Read())
        {
            CUneEtape oUneEtape = new CUneEtape(oModule.voBase);
            lstEtapes.Add(oUneEtape);
        }
    }
    catch (Exception vEx)
    {
        oModule.OnError(vEx);
    }

    finally { oModule.voBase.Close(); }

    return lstEtapes;
}
```

Une liste est alors retournée.

La classe CFLUX_Questionnaire va quant à elle se charger de parcourir cette liste et sérialiser les données pour construire étape après étapes le flux souhaité.

```
private string GetFluxEtapes()  //Retourne le Flux d'initialisation de la ProgressBar
{

    string sFlux = "";
    _nIdEtapeEnCours = (short)_lstEtapes[0].ID_ETAPE;

    foreach (CQuestionnaire.CUneEtape oUneEtape in _lstEtapes) //Construction du Flux
    {
        sFlux += oUneEtape.LIB_ETAPE + SEPARATEUR_FLUX.m_FLUX_SEPARATEUR_DATA_FIELDS
              +  oUneEtape.TYP_ETAPE.GetHashCode() + SEPARATEUR_FLUX.m_FLUX_SEPARATEUR_DATA_OBJECT;
    }

    return sFlux;
}
```

Ensuite de retour dans le fichier JS, il suffit de récupérer le flux correspondant à la barre de progression.

```
function OnInitialiseDone(pi_sResult){
    var oHWCResult = HWResult(pi_sResult);
    if(oHWCResult.vbOK){

        //Flux d'init de la progress bar
        var sFluxInitPGB = oHWCResult.voParam.vTabResult(0);
```

Puis ensuite, comme précédemment appelé notre objet CPPH en envoyant notre flux généré dynamiquement.

```
//Objet globale de la page fquestionnaire
g_oPPH = new CPPH(sFluxInitPGB,sFluxEtapeEnCours,sFluxInitEtape,sFluxMainPage);
```

Compétences du référentiel au cours de cette semaine :

A1.4.1
Participation à un projet
A4.2.1
Analyse et correction d'un dysfonctionnement, d'un problème de qualité
C4.1.6.1
Mettre en place et exploiter un environnement de développement
1.4.1.2
Rendre compte de son activité
C4.1.2.2
Maquetter un élément de la solution applicative
C4.1.5.2
Développer un prototype
C4.1.6.2
Mettre en place et exploiter un environnement de test

A4.1.1
Proposition d'une solution applicative (Mise en place de la bdd)