

Rapport de Stage

Fleuron d'Anjou

Sommaire :

<u>Présentation</u>	2
<i>L'entreprise :</i>	2
<i>Le stage :</i>	2
Mon tuteur	2
Le Projet	2
<u>Missions</u>	3
<i>Première mission :</i>	3
<i>La vue</i>	3
<i>Le contrôleur</i>	4
<i>Bilan</i>	4
<i>Deuxième mission :</i>	5
<i>Les vues</i>	5
<i>Le contrôleur et le modèle</i>	6
<i>Bilan</i>	7
<u>Projet</u>	8
<i>Première tâche :</i>	8
<i>La vue</i>	8
<i>Le contrôleur</i>	9
<i>Bilan</i>	10
<i>Deuxième tâche :</i>	11
<i>Le Widget</i>	11
<i>Le Contrôleur</i>	11
<i>La vue</i>	12
<i>Bilan</i>	12
<i>Troisième tâche :</i>	13
<i>Le Widget</i>	13
<i>Le Contrôleur</i>	15
<i>La vue</i>	16
<i>Bilan</i>	16
<u>Conclusion</u>	17

Présentation

L'entreprise :

Nom : Fleuron d'Anjou

Adresse : 29 Avenue du Moulin Marcille,
49136 Les Ponts-de-Cé

Téléphone : 02 41 96 66 66

Site internet : <https://www.fleurondanjou.fr/>



Description : Fleuron d'Anjou est une coopérative maraîchère et horticole créée en 1962 aux Ponts-de-Cé par des maraîchers de la région d'Angers et regroupant près de 80 producteurs horticoles et maraîchers et 298 employés sur 6 sites. La coopérative s'occupe de vendre les produits aux jardinerie, aux grossistes, aux grandes surfaces ...

Le stage :

Mon tuteur

Lors de ce stage, je suis supervisé par M. Antoine JANVRIN.

M. JANVRIN est le DSI de la coopérative Fleuron d'Anjou.

Je suis aussi accompagné par Yoann BOURDEAU, membre de l'équipe de développement, qui me suis au quotidien et m'aide lors de mes différentes tâches.

Le Projet

Lors de ce stage, je suis assigné à l'équipe de développement du groupe, qui a pour objectif d'assurer le développement et la maintenance des suites logicielles, assurer le support aux utilisateurs, garantir l'intégrité des données, assurer les plans d'activité, assurer le bon fonctionnement des télécommunication ...

Mon projet va se dérouler sur l'application Web Gaïa développée par et pour le groupe Fleuron d'Anjou. Gaia est une GPAO (Gestion de Production Assisté par Ordinateur), elle permettra la saisie des prévisions de vente au travers d'un PDP (Plan Directeur de Production) qui déclenchera le CBN (Calcul des Besoins Net) pour permettre aux adhérents de la coopérative d'avoir un rétroplanning de leurs besoins.

L'application utilise le Framework PHP YII2 pour ses performances, sa sécurité et son aspect modulable notamment grâce à ses widgets.

Quant à la base de données, elle est hébergée sous Oracle et est utilisée avec des vues.

Missions

Première mission :

Cette première mission a pour but de me faire découvrir l'architecture de l'application Gaïa et le Framework Yii2. Il fallait que je fasse une refonte de la page profil utilisateur afin de la rendre plus lisible et avec plus d'information.

La page a pour seul but d'afficher les informations de l'utilisateur. Pour modifier ces informations, l'utilisateur aura l'accès à une autre page en fonction de ses droits.

Base de données TEST	
X Courapié Brieuc	
Code Pointage	1234
Nom	Courapié
Prenom	Brieuc
Genre	H
Service	Informatique
Poste	Développeur
Email	brieuc.courapie@fleurondanjou.fr
Tel interne	
Tel ligne directe	
Tel portable	
Status	Active
Roles	
Backend écriture	
Frontend écriture (admin)	Admin

Page déjà existante

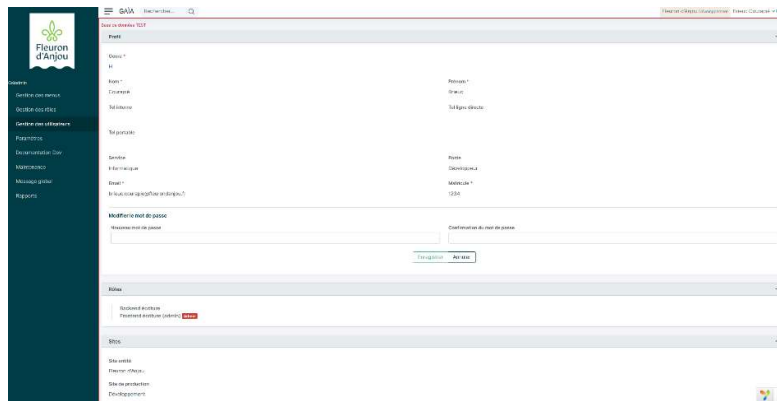
La vue

Page modification de l'utilisateur

J'ai donc recréé une vue et ai repris le code de la page de modification de l'utilisateur afin de le rendre non modifiable. Pour cela, il a fallu rendre les champs seulement lisibles.

```
<div class="col-12 col-xl-6">
  {{ form.field(user, 'SNOM').textInput({'class':'form-control-plaintext','readonly':true}) | raw }}
</div>
```

Le `form.field` est un élément du widget `active-form` qui permet de créer un champ de formulaire automatiquement en lui rentrant en paramètre un modèle (`user`) la valeur qui le définit (`SNOM`). L'application utilisant le Framework CSS Bootstrap j'ai donc mis la classe `form-control-plaintext` pour ne pas avoir de rebord sur les inputs ainsi qu'un `readonly` pour ne pas pouvoir changer le texte.



Résultat final de la vue

Le contrôleur

Concernant le contrôleur, on commence par aller chercher toutes les données de l'utilisateur. On affiche ensuite la vue en y rentrant toutes les données ainsi que des liens.

L'utilisateur ayant la possibilité de changer son mot de passe dans la vue, il faut donc le traiter. Pour cela, on charge N_SMDP et C_SMDP en post qui sont le nouveau mot de passe rentré par l'utilisateur. Ensuite on appelle la fonction `savePassword()` du modèle `V_USR_G` qui vérifie que les mots de passes rentrés sont identiques et qui met à jour le mot de passe dans la base de données.

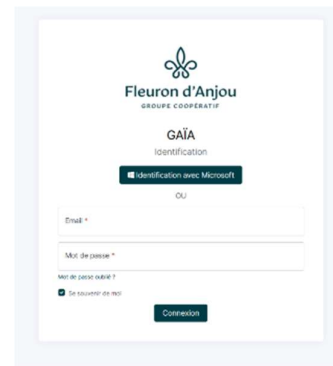
```
public function actionProfileUser($IDUSR)
{
    // $user = USR::searchById($IDUSR);
    // $user = V_USR_G::searchById($IDUSR);
    $roles = V_USR_ROLE::findAll(['IDUSR' => $IDUSR]);
    $usrSite = new V_USR_SITE();
    $siteSearched = V_USR_SITE::listeById($IDUSR);
    if ($siteSearched != null) {
        $usrSite = $siteSearched[0];
    }
    // changement du model avec les valeurs post si elles existent
    $newUser = new V_USR_G();
    if ($newUser->load(Yii::$app->request->post())) {
        if (empty($newUser->N_SMDP)) {
            $user->N_SMDP = $newUser->N_SMDP;
            $user->C_SMDP = $newUser->C_SMDP;
            // validation et sauvegarde du nouveau mot de passe
            if ($user->savePassword()) {
                Yii::$app->session->addFlash(
                    'success',
                    'Enregistrement du mot de passe réussi !'
                );
                return $this->redirect(Yii::current());
            } else {
                Yii::$app->session->addFlash(
                    'danger',
                    'Echec d\'enregistrement du mot de passe !' .
                    Html::errorSummary($user)
                );
            }
        } else {
            Yii::$app->session->addFlash(
                'warning',
                'Nouveau mot de passe vide !'
            );
        }
    }
    return $this->render('fiche-generale.html.twig', [
        'user' => $user,
        'roles' => $roles,
        'genres' => USR::$genres,
        'url_fiche' => Url::to(['user/fiche', 'IDUSR' => $user->IDUSR]),
        'url_roles' => Url::to(['user/fiche-roles', 'IDUSR' => $user->IDUSR, 'tab' => 'roles']),
        'url_sites' => Url::to(['user/fiche-sites', 'IDUSR' => $user->IDUSR, 'tab' => 'roles']),
        'url_supprimer' => Url::to(['user/supprimer', 'IDUSR' => $user->IDUSR]),
        'sens' => V_SEN_ALL::liste(),
        'sprs' => V_SPR::listeSpr(),
        'sves' => V_SVE::liste(),
        'usrSite' => $usrSite,
    ]);
}
```

Bilan

Cette mission m'a permis de découvrir le Framework Yii2 ainsi que l'application Gaïa. J'ai pu y explorer l'architecture du site et sa manière de fonctionner. C'était une des missions les moins complexe de mon stage étant donné que j'ai principalement eu besoin de reprendre du code existant ailleurs et de l'agencer de la bonne manière. Cette tâche est ensuite partie en phase de test puis en production.

Deuxième mission :

Cette mission a pour objectif la création du système de gestion du mot de passe oublié dans le formulaire de connexion de Gaïa.



Les vues

Pour commencer, il faut créer la vue permettant à l'utilisateur de rentrer son adresse email, afin qu'un email lui soit envoyé pour pouvoir réinitialiser son mot de passe.



Vue pour l'envoi d'email

Il faut ensuite créer le mail qui va contenir le lien pour réinitialiser son mot de passe.

A savoir que dans un email HTML, les balises `<style>` ne fonctionnent pas et donc fait apparaître le code CSS. Pour styliser l'email, il faut donc utiliser l'attribut `style` dans les balises.



Email reçu par l'utilisateur

Pour finir, j'ai créé la vue permettant à l'utilisateur de changer son mot de passe. Avec deux entrées pour que l'utilisateur saisisse bien son mot de passe.



Vue de changement de mot de passe

Le contrôleur et le modèle

La méthode `actionRequestPasswordReset()` dans le contrôleur, affiche la vue pour l'envoi d'email en reprenant le layout de la page de login. Lors de l'envoi du formulaire, on récupère la requête en post, puis on exécute la fonction `sendEmail()` du modèle. Pour finir, on prévient l'utilisateur.

```
/**
 * Requests password reset.
 *
 * @return mixed
 */
public function actionRequestPasswordReset()
{
    $this->layout = 'login';
    $model = new PasswordResetRequestForm();
    if ($model->load(Yii::$app->request->post())){
        if ($model->validate()) {
            if ($model->sendEmail()) {
                Yii::$app->session->setFlash('success', 'Vérifiez vos emails pour plus d\'instructions.');
                return $this->goHome();
            }

            Yii::$app->session->setFlash('error', 'Désolé, nous ne pouvons pas réinitialiser votre mot de passe pour l\'adresse email fournie.');
        }
    }
    return $this->render('requestPasswordResetToken.html.twig', [
        'title' => 'Réinitialisation du mot de passe',
        'model' => $model,
    ]);
}
```

La fonction `sendEmail()` permet de gérer l'envoi d'Email via la base de données Oracle. Pour cela, On commence par vérifier si l'email entré est présent en base de données. Ensuite, on génère un token unique permettant de sécuriser l'URL. Puis, on crée le message de l'email qui sera un fichier HTML avec le lien vers la page de réinitialisation du mot de passe et des données de l'utilisateur. On crée la requête SQL avec : les emails de l'expéditeur et du destinataire, l'objet du mail et le message. Pour finir, on exécute la commande.

```
public function sendEmail()
{
    /* @var $user User */
    $user = Yii::$app->user->findOne([
        'email' => $this->email,
    ]);
    if (!$user) {
        return false;
    }
    $user->generatePasswordResetToken();
    $this->MESSAGE = Yii::$app->controller->renderPartial('mailResetPassword.html.twig', [
        'resetUrl' => Yii::$app->urlManager->createUrl(['site/reset-password', 'IDUSR' => $user->IDUSR, 'token' => $user->password_reset_token]),
        'user' => $user,
    ]);
    //utilisation d'une fonction Oracle pour l'envoi d'email
    $command = Yii::$app->db->createCommand("
        DECLARE
            P_EXP VARCHAR2(200);
            P_DEST VARCHAR2(200);
            P_OBJ VARCHAR2(200);
            P_MSG VARCHAR2(1000);

        BEGIN
            P_EXP := :P_EXP;
            P_DEST := :P_DEST;
            P_OBJ := :P_OBJ;
            P_MSG := :P_MSG;

            MAIL.SEND_MAIL(
                P_EXP => P_EXP,
                P_DEST => P_DEST,
                P_OBJ => P_OBJ,
                P_MSG => P_MSG
            );
            --rollback;
        END;");

    $command->bindValue(':P_EXP', Yii::$app->params['adminEmail'], PDO::PARAM_STR);
    $command->bindValue(':P_DEST', $this->email, PDO::PARAM_STR);
    $command->bindValue(':P_OBJ', 'Réinitialisation du mot de passe - Gaia', PDO::PARAM_STR);
    $command->bindValue(':P_MSG', $this->MESSAGE, PDO::PARAM_STR);
    $res = true;
    try {
        if (!$command->execute())
            $res = false;
    } catch (Exception $e) {
        $res = false;
    }
    return $res;
}
```

Fonction SendEmail dans le modèle PasswordResetRequestForm

La méthode *actionResetPassword()* dans le contrôleur, affiche la vue pour changer son mot de passe en reprenant le layout de la page de login. Pour commencer, la fonction vérifie si le token présent dans l'URL est correct et autorise ou non l'accès à la page. Ensuite, on vérifie les mots de passes entrés et on les sauvegarde dans la base de données avec la méthode *savePassword()* du modèle *ResetPasswordForm()*. Pour finir, on prévient l'utilisateur que l'enregistrement a eu lieu ou non avec un message.

```
public function actionResetPassword($IDUSR, $token)
{
    $this->layout = 'login';

    try {
        $model = new ResetPasswordForm($IDUSR, $token);
    } catch (InvalidArgumentException $e) {
        throw new BadRequestHttpException($e->getMessage());
    }
    $newUser = new V_USR_G();
    if ($newUser->load(Yii::$app->request->post())) {

        if (empty($newUser->N_SMDP))
        {
            $model->_user->N_SMDP = $newUser->N_SMDP;
            $model->_user->C_SMDP = $newUser->C_SMDP;

            //validation et sauvegarde du nouveau mot de passe
            if ($model->_user->savePassword()) {
                Yii::$app->session->addFlash(
                    'success',
                    'Enregistrement du mot de passe réussi !'
                );
                return $this->redirect(Url::to(['login']));
            } else {
                Yii::$app->session->addFlash(
                    'danger',
                    'Echec d\'enregistrement du mot de passe !' .
                    Html::errorSummary($model->_user)
                );
            }
        } else {
            Yii::$app->session->addFlash(
                'warning',
                'Nouveau mot de passe vide !'
            );
        }
    }

    return $this->render('resetPassword.html.twig', [
        'model' => $newUser,
        'title' => 'Changement de mot de passe'
    ]);
}
```

Bilan

Cette mission m'a permis de continuer d'apprendre le Framework et le l'application Gaïa. Il a été notion de cybersécurité lors de la création des pages ce qui m'a permis de réfléchir à comment sécuriser les liens entre les différentes pages notamment avec la création et la comparaison d'un token. J'ai aussi appris à réaliser un mail en HTML, celui-ci m'a posé des problèmes avec le style du mail qui devait être avec du CSS classique, j'ai donc dû m'adapter. La fonctionnalité a ensuite été mise en production.

Projet

Lors de mon projet, je vais travailler sur la partie des étapes de production. Dans Gaïa, chaque article peut posséder une ou plusieurs étapes de production. Ces étapes possèdent des temps de réalisation, néanmoins, les temps de chaque étape peuvent varier d'une semaine à une autre.

L'objectif de mon projet sera donc de rendre ces durées plus simples à lire et à implémenter pour l'utilisateur.

Première tâche :

Avant de commencer, Gaïa possède déjà un modal (fenêtre contextuelle bloquant l'interaction avec la page) pour rentrer les durées des étapes, néanmoins, celle-ci n'est pratique à l'utilisation car elle affiche les étapes seulement semaine par semaine.

Étape de production	Durée	Semaine de réalisation	Durée modifiée
Repiquage 1	5	Semaine n°48 N-1	
Distancage	1	Semaine n°49 N-1	
Cumul	0	Semaine n°1	

Nous avons donc réfléchi à comment rendre plus simple la saisie de données pour l'utilisateur. Lors d'un brainstorming, nous avons trouvé que le plus pratique pour l'utilisateur et le plus simple à implémenter serait un tableau modifiable à double entrées avec les semaines en ligne et les étapes en colonne dans une modal.

La vue

Pour commencer, le tableau va être réalisé avec le widget *GridViewGaia* (créé pour l'application Gaïa), ce qui va me permettre de facilement le paramétrer. De plus, le widget va me permettre de mettre un formulaire de saisie à l'intérieur du tableau.

Semaine	Repiquage 1	Cumul	
1	20	1	✓ ↺
2	20	1	
3	20	0	

Formulaire de saisie du tableau

Durée manuelle			
Article : ABELIA KALEIDOSCOPE® P2L			
Site de Production : Production			
Semaine	Repiquage 1	Cumul	
1	20	1	
2	20	0	
3	20	0	
4	20	0	
5	20	0	
6	20	0	
7	20	0	
8	20	0	
9	20	0	
10	20	0	
11	20	0	
12	20	0	
13	20	0	
14	20	0	
15	20	0	
16	20	0	
17	20	0	

Rendu final du tableau

La construction d'un tableau avec le widget *GridViewGaia* se fait de la manière ci-dessous, avec les différents paramètres. Par exemple : *modelePk* permet d'indiquer la clé primaire du tableau, *dataProvider* permet d'insérer les données du tableau, *gvEditable* permet d'autoriser ou non la modification des champs, *columns* permet d'indiquer les différentes colonnes à afficher...

```
{ { grid_view_gaia_widget({
    'gvEditable' : true,
    'gvPager' : false,
    'id' : 'grid-sem-etp',
    'dataProvider' : etpSemDp,
    'tableOptions' : {'class' : 'gaia-table gaia-table-hover gaia-table-sticky-thead table-level-1'},
    'modelePk' : 'NUANNSEMV',
    'type' : 'line',
    'pjaxId' : 'pjax-sem-etp',
    'form' : form,
    'actionsRights' : true,
    'showFirstCol' : true,
    'showSummary' : false,
    'modal' : false,
    'columns' : gvColumns
  }) | raw
}}
```

```
{% set gvColumns = [
  {
    'attribute' : 'NUANNSEMV',
    'label' : 'Semaine',
    'value' : week,
  }
]
}%

{% for etp in etps %}
  {
    set gvColumns = gvColumns|merge([
      {
        'attribute' : 'DUREE' ~ etp.IDETP,
        'label' : etp.LLETP,
        'format' : 'raw',
        'options' : {
          'type' : 'number',
          'editable' : true,
          'width' : '100px',
        },
        'contentOptions' : etpCellOptions,
      }
    ])
  }
{% endfor %}
```

Ensuite, il faut préciser au widget les colonnes à faire apparaître. Premièrement, la colonne semaine va faire apparaître les 53 semaines de l'année, nous passons donc en paramètre de la colonne les données de NUANNSEMV. Néanmoins, les données de NUANNSEMV sont de format « 202001 » et pour avoir seulement le numéro de la semaine, nous rentrons en valeur la variable *week* que nous verrons dans le contrôleur.

De plus, chaque article possède un nombre différent d'étapes, on utilise alors une boucle For pour ajouter les colonnes d'étapes en fonction de leur nombre.

Le contrôleur

Pour commencer, le contrôleur récupère l'article avec son ID, puis, les étapes de production de l'article et le dataProvider (Ensemble des données du tableau).

Ensuite, pour chaque étape de l'article, on instancie les *rules* qui permettent, pour le formulaire de saisie, que l'input soit un entier positif et non-nul. On instancie aussi le label de chaque étape pour l'affichage des erreurs.

```
$article = VA_ART_ALL::findOne(['IDART' => $IDART]);
if(empty($article))
    throw new GoneHttpException("L'article est introuvable ou inexistant !");

$etps = VA_ART_ETP::getEtpsById($IDART, $IDSPR);
$etpSemDp = VA_ART_ETP_SEM::listeGroupeDp(['F_IDART' => $IDART, 'F_IDSPR' => $infosSpr['site']->IDS]);

//renseignement des rules et labels des colonnes DUREE (dynamiques)
foreach($etps as $etp){
    VA_ART_ETP_SEM::$customRules[] = ['DUREE'. $etp->IDETP, 'integer', 'min' => 0];
    VA_ART_ETP_SEM::$customRules[] = ['DUREE'. $etp->IDETP, 'required'];
    VA_ART_ETP_SEM::$customAttributeLabels['DUREE'. $etp->IDETP] = $etp->LLETP;
}
```

Ensuite, la variable `$etpCellOptions` permet de modifier la forme des cellules une par une pour savoir si la donnée a déjà été modifiée par un utilisateur. Si c'est le cas, alors la fonction va retourner une classe qui permet de griser la cellule avec une barre verticale.

Ensuite, la variable `$renderWeek` permet d'afficher seulement le numéro de la semaine au lieu de la valeur du `nuannsemv` qui est de format « 202001 ». Pour cela, on prend les deux derniers nombres de `nuannsemv` et on les transforme en entier. Puis, on envoie la valeur dans une vue avec seulement une variable.

```
$etpCellOptions = function ($data, $x, $y, $column) {
    $class = "";
    $idetp = substr($column->attribute, 5);
    $etpsem = current(array_filter($data->durees, function($duree) use($idetp){
        return $duree->IDETP == $idetp;
    }));

    if($etpsem->VADUREEM != null /*&& $etpsem->VADUREE != $etpsem->VADUREEM*/)
    {
        $class = 'gaia-table-active lborder-dark'; //ajout d'une bordure épaisse à gauche
    }
    return ['class' => $class];
};

$renderWeek = function($data){
    return $this->renderPartial('/article/article-prm-prod/sem-etp/sem.html.twig', [
        'week' => intval(substr($data->NUANNSEMV, -2))
    ]);
};

return $this->renderAjax('/article/article-prm-prod/sem-etp/_modal-sem-etp2.html.twig', [
    'etpSemDp' => $etpSemDp,
    'etps' => $etps,
    'etpCellOptions' => $etpCellOptions,
    'week' => $renderWeek,
    'urlModifierEtpSem' => Url::to(['ajax-modifier-etp-sem', 'IDSPR' => $IDSPR, 'IDART' => $IDART])
    // 'renderDuree' => $renderDuree,
]);
```

Bilan

Cette tâche m'a permis d'appréhender les widgets qui seront essentiel pour la suite du projet. J'ai pu bénéficier de la doc réalisée par l'équipe afin de paramétrer à ma guise le tableau. Le plus compliqué dans cette tâche, a été tout ce qui concerne les données. La base de données n'étant pas organisé de la bonne manière pour inscrire les données dans le tableau, nous avons commencé par créer une nouvelle vue SQL avec un fonction PIVOT pour se raviser et utiliser l'ORM du Framewok car plus simple à utiliser par la suite.

Deuxième tâche :

Dans Gaïa, les semaines ont la possibilité d'être regroupées afin de simplifier le nombre d'étapes.

L'objectif est donc d'implémenter la fonctionnalité dans notre modal

Le Widget

Pour implémenter la fonctionnalité dans le tableau des semaines, il va falloir ajouter un paramètre dans le widget *GridViewGaia* afin de ne lire seulement la première ligne de semaine regroupées. Cela va permettre d'afficher une seule ligne par regroupement.

Pour cela, on ajoute le paramètre *visibleCondition* dans notre widget.

```
/**
 * Fonction (renvoyant un boolean) permettant de déterminer si une ligne est visible ou non
 * @var [Function]
 */
public $visibleCondition = null;
```

Ensuite, dans la fonction déjà existante permettant d'afficher les lignes d'un tableau, on ajoute le code ci-dessous. Celui-ci commence par vérifier si le paramètre a été instancié. Si c'est le cas, on appelle le résultat de la fonction dans le contrôleur qui déterminera si une ligne doit être affichée ou non, ce qui retournera un booléen

Pour finir, on retourne la vue en fonction du résultat obtenue pour chaque ligne.

```
//Vérification si une condition d'affichage de la ligne existe
$isRowVisible = true;
if(!empty($this->visibleCondition) && $this->visibleCondition instanceof Closure )
    $isRowVisible = call_user_func($this->visibleCondition, $model, $key, $index);
//---
if($isRowVisible)
    return $this->gvEditable ? $this->renderTableRowGve($model, $key, $index, $optionsRow = [], $optionsInput = []) : parent::renderTableRow($model, $key, $index);
else
    return '';
```

Le Contrôleur

Nous allons ici modifier le contrôleur précédemment utilisé pour afficher la modal. Dans le modèle, nous avons ajouté la sélection de la colonne GRP qui représente le groupement de semaine. Par exemple, si les semaines 1 à 10 sont regroupées, alors, leurs tuples GRP seront « 01-10 », à l'inverse, si la ligne 11 n'est pas regroupée alors son tuple GRP sera « 11-11 ».

```
$rowVisibleCondition = function($data){
    if($this->lastRowVal == $data->GRP){
        $this->lastRowVal = $data->GRP;
        return false;
    }else
    {
        $this->lastRowVal = $data->GRP;
        return true;
    }
};
```

Pour pouvoir savoir si une ligne doit être affichée ou non, on crée la variable booléenne *\$rowVisibleCondition* ainsi que la variable *\$lastRowVal* qui va nous permettre de stocker la valeur de la ligne précédente. Pour chaque ligne, on vérifie si sa donnée GRP est égale à la donnée GRP de la ligne précédente, si c'est le cas cela veut dire que les deux lignes sont regroupées alors, on ne l'affichera pas.

Pour finir, la variable est envoyée dans la vue.

La vue

Peu de chose ont besoin d'être modifié sur la vue. On rajoute évidemment le paramètre *visibleCondition* qu'on instancie dans la construction de notre tableau GridViewGaia.

Nous allons aussi afficher dans le tableau, le regroupement au lieu de la semaine. Pour cela, nous allons reprendre la variable *renderWeek* du contrôleur. On commence par vérifier si une ligne est groupée en comparant les deux chiffres dans GRP, si ce sont les mêmes, alors la ligne n'est pas regroupée et l'on affiche seulement un chiffre sino on affiche le regroupement.

```
$renderWeek = function($data){
    if(substr($data->GRP, 0, 2) == substr($data->GRP, -2)){
        $week = intval(substr($data->GRP, 3));
    }
    else{
        $week = $data->GRP;
    }
    return $this->renderPartial('/article/article-prm-prod/sem-etp/sem.html.twig', [
        'week' => $week
    ]);
};
```

Durée manuelle

Article : **AB AGAPANTHE L100/110 BLEU P24**

Site de Production : Production

Semaine	Repliquage 1	Distancage	Cumul
50-03	5	1	0
4	5	1	0
05-10	5	1	0
11	4	1	0
12	4	1	0
13	4	1	0
14	3	1	0
15	3	1	0
16	3	1	0
17	3	1	0
18	3	1	0
19	3	1	0
20	3	1	0
21	2	1	0
22	2	1	0
23	2	1	0
24	2	1	0

Rendu final de la vue

Bilan

Cette tâche était donc une version améliorée de la précédente. Le plus compliqué ici étant d'ajouter un paramètre dans le widget.

Troisième tâche :

Cette troisième tâche doit permettre à l'utilisateur de visualiser la durée des étapes en fonction des différents regroupements afin d'avoir un rétroplanning. Nous avons décidé de réaliser un graphique à barres flottantes qui permettra de voir le début et la fin de chaque étape. Pour réaliser ce graphique, j'ai choisi d'utiliser la librairie Javascript Chart.js car c'est une des seules qui peut réaliser notre demande ainsi qu'une librairie très utilisée et donc très documentée. Afin que l'on puisse, à l'avenir créer d'autres graphiques dans l'application, nous allons créer un nouveau widget.

Le Widget

Nous allons commencer par créer le widget. Il recevra les données du contrôleur en PHP et les enverra au format JSON au JavaScript. Pour commencer, on instancie les différents paramètres du widget, on crée la fonction *init()* qui va permettre d'envoyer de convertir les données au format JSON et de l'envoyer au JavaScript, puis on crée la fonction *run()* qui affichera le graphique.

Ensuite, la fonction *getChartData()* va permettre de mettre en ordre les données afin que la librairie puisse les lire. Nous allons donc créer un dataset (ensemble des données) par étapes avec à l'intérieur, le nom de chaque étape, son tableau de données, sa largeur, sa couleur sachant qu'une étape correspond à une couleur prédéfinie, son type ...

Pour finir, les datasets sont envoyés dans un tableau avec les labels qui sont les différents regroupements de l'article.

```
public function getchartDatas()
{
    $datasets = [];
    $color = [ "CM" => '#fcb92c', "ENT" => '#c83c3c', "DS" => '#6c757d',

    foreach ($this->datasets as $value) {
        if(!empty($this->datas[$value[$this->datasetAttribute]]))
        {
            $data = $this->datas[$value[$this->datasetAttribute]];
        }
        $datasets[] = [
            "label" => $value[$this->labelAttribute],
            "data" => $data,
            "fill" => false,
            "borderColor" => $color[$value['COETP']],
            "backgroundColor" => $color[$value['COETP']],
            "type" => $this->type,
            "stack" => 'stack 1',
            "barPercentage" => $this->options['width'],
        ];
    }

    $datas = array(
        "labels" => $this->labels,
        "datasets" => $datasets,
    );

    return $datas;
}
```

```
// Options pour le graphique
const options = {
  scales: {
    beginAtZero: false,
    yAxes: [{
      stacked: true,
      ticks: {
        reverse: true,
        stepSize: 1,
      },
      scaleLabel: {
        display: true,
        labelString: 'Regroupement des semaines',
      },
    }],
    xAxes: [{
      scaleLabel: {
        display: true,
        labelString: 'Durée des étapes',
      },
      ticks: {
        callback:
          function(value, index, ticks) {
            return 'S ' + ((value % 52)+1);
          },
        suggestedMin: 0,
        suggestedMax: 103,
        stepSize: 2,
      },
    }],
  },
};
```

Dans la partie JavaScript, on commence par récupérer les données du PHP puis on crée les options de mise en page de notre graphique. On commence par configurer les axes X et Y. L'axe Y correspondra aux regroupements des semaines avec, pour chaque regroupement, les différentes étapes. L'axe X correspondra aux durées des étapes représentées en semaine. Le rétro planning pouvant se tenir sur deux années différentes, on appliquera donc un modulo 52 afin d'avoir des années de 52 semaines. Pour éviter d'avoir une semaine 0 on ajoute 1 aux valeurs de l'axes mais aussi dans les données que l'on formatera dans le contrôleur.

```
title: {
  display: true,
  text: "Rétro-planning des étapes de productions",
},
tooltips: {
  callbacks: {
    label: function(tooltipItem, data) {
      var value = data.datasets[tooltipItem.datasetIndex].data[tooltipItem.index];
      if(tooltipItem.datasetIndex == 0){
        return 'Début étape : semaine ' + ((value[0] % 52)+1);
      }
      else{
        var retour = data.datasets[tooltipItem.datasetIndex-1].data[tooltipItem.index][1];
        for(var i = 1 ; i < tooltipItem.datasetIndex ; i++){
          retour += data.datasets[i-1].data[tooltipItem.index][1];
        }
        return 'Début étape : semaine ' + ((retour % 52)+1);
      }
    }
  },
}
```

Ensuite, on ajoute un titre au graphique et pour finir, on va configurer les tooltips. Les tooltips permettent d'afficher les données à l'utilisateur lorsque celui-ci survole le graphique avec sa souris. Pour cela, on récupère les données puis on les formate modulo 52 plus 1. Le format des données de la première étape et des autres étant différent, on les traite donc différemment.

```
// Créer le graphique
const ctx = $(this)[0].getContext('2d');
const myChart = new Chart(ctx, {
  type: 'horizontalBar',
  data: dataset,
  options: options
});
```

Pour terminer, on envoie le dataset et les options dans le graphique.

Le Contrôleur

Afin de créer le graphique, les données ont besoin d'être parfaitement formaté pour que le javascript affiche le graphique que l'on veut. Nous allons ici réutiliser la fonction affichant la modal utilisé précédemment.

On commence par récupérer les différents regroupements uniques des étapes.

Puis, on sépare les semaines par étapes en les mettant dans un tableau.

```
//Sélectionne la durée des étapes pour chaque semaine
foreach ($setps as $setp) {
    $stop = "";
    for($i = 0; $i < 52; $i++) {
        if($stop != $preDats[$setp->IDETP][$i]->GRP){
            $stop = $preDats[$setp->IDETP][$i]->GRP;
            if($preDats[$setp->IDETP][$i]->VADUREEM){
                $dats[$setp->IDETP][$stop] = $preDats[$setp->IDETP][$i]->VADUREEM;
            }
            $dats[$setp->IDETP][$stop] = $preDats[$setp->IDETP][$i]->VADUREE;
        }
    }

    //calcul du temps total de chaque semaine de vente
    foreach($grp as $i)
    {
        $a = 0;
        foreach($setps as $setp){
            $a += $dats[$setp->IDETP][$i];
        }
        $dureeSem[$i] = $a;
    }
}
```

```
//Classe tout les regroupements dans un tableau
foreach($setps as $setp){
    $groupe = $setp->GRP;
    if($g != $groupe){
        $g = $groupe;
        array_push($grp, $groupe);
    }
}

//Classe toute les étapes dans un tableau
foreach($setps as $setp){
    $idstp = $setp->IDETP;
    $preDats[$setp->IDETP] = array_filter($setps, function($elt) use($idstp) {
        return $elt->IDETP == $idstp;
    });
    $preDats[$setp->IDETP] = array_values($preDats[$setp->IDETP]);
}
```

Ensuite, pour chaque étape, on met en tableau la durée d'étape avec le regroupement en index pour ne pas avoir de répétition pour les semaines regroupées.

Puis, on calcule la durée totale des étapes pour chaque semaine regroupement.

Ensuite, on met en forme le tableau des données sachant que pour avoir des barres sans commencer par zéro sur le graphique, une donnée doit avoir un début et une fin, donc un tableau de deux valeurs. Autre particularité, seulement la première étape a besoin d'un début, les étapes suivantes ont besoin d'une seule donnée. Donc pour que la première étape soit bien placée sur le graphique, on ajoute 52 ce qui correspond au début de l'année, puis on retire le temps total des étapes, puis on ajoute le numéro de la semaine finale et on rajoute 1 car on avait fait -1 dans le widget pour ne pas avoir de semaine 0. Pour les étapes suivantes, on met seulement la durée de l'étape.

```
//Ordonne les données pour le graphique
$dataset = array();
foreach($setps as $setp){
    $i = 0;
    foreach($dats[$setp->IDETP] as $data)
    {
        if($nbEtp == 1)
        {
            $dataset[$setp->IDETP][$i] = [52-$dureeSem[$grp[$i]]+intval(substr($grp[$i], 0,2))-1, 52-$dureeSem[$grp[$i]] + $data + intval(substr($grp[$i], 0,2))-1];
        }
        else
        {
            $dataset[$setp->IDETP][$i] = [0, $data];
        }
        $i++;
    }
    $nbEtp++;
}
```



```
//Création des labels pour le graphique (Axes Y)
$labels = [];
foreach ($grp as $value) {
    if(substr($value, 0, 2) == substr($value, -2)){
        $value = strval(intval(substr($value, 3)));
    }
    array_push($labels, $value);
}

// Envoi la largeur des barres
$width = 1;
if(count($grp) == 1){
    $width = 0.25;
}elseif(count($grp) > 1 && count($grp) < 5){
    $width = 0.5;
}
```

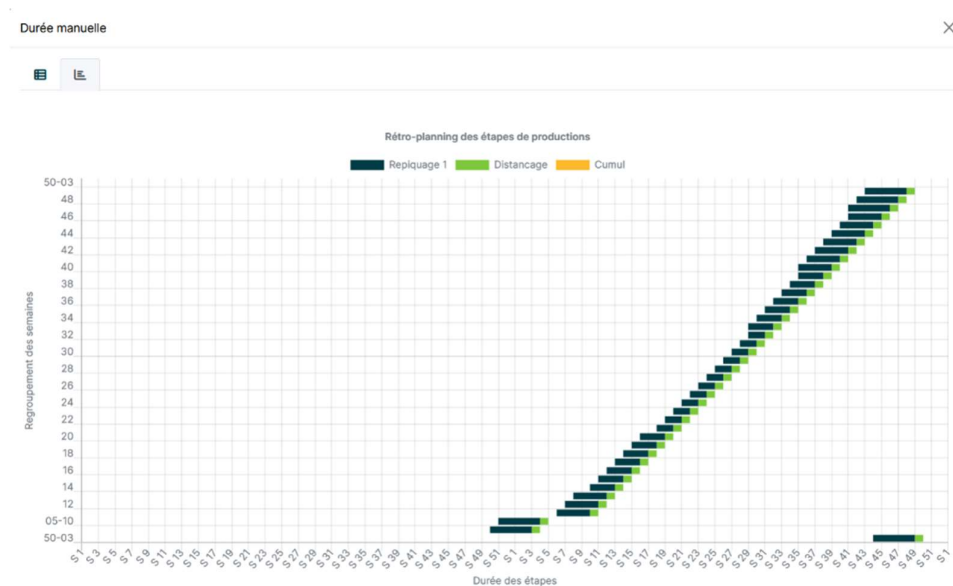
Pour finir, on crée les labels pour l'axe Y avec les regroupements, et on modifie la largeur des barres en fonction de leur nombre dans le tableau pour qu'il soit plus lisible lorsqu'il y a peu de regroupements.

La vue

Étant donné que nous utilisons la même modal précédemment utilisé pour le tableau, nous avons ajouté des onglets liste et graphique pour mieux naviguer.

Il ne reste plus qu'à appeler notre widget dans notre vue en y passant les paramètres créés dans le contrôleur ainsi que d'autres comme le type ou les différents attributs.

```
{{ chart_widget({
    'chartId' : 'etpsem',
    'type' : 'horizontalBar',
    'datasets' : etps,
    'datas' : datas,
    'datasetAttribute' : 'IDETP',
    'labelAttribute' : 'LLETP',
    'labels' : labels,
    'options' : {
        'width' : width,
        'stepped' : true,
    }
}) | raw
}}
```



Rendu final du graphique

Bilan

Cette mission était de loin la plus longue et la plus compliquée du stage. En effet, il a été compliqué de trouver comment s'y prendre pour avoir ce résultat, entre : comment bien formater les données sans avoir de soucis d'affichage, bien paramétrer le graphique pour que les axes soient bons, retourner les données utiles à l'utilisateur...

Conclusion

Pour conclure, ce stage m'a permis de gagner en expérience dans le monde professionnel ainsi que d'approfondir mes connaissances sur la programmation en entreprise.

J'y ai découvert un nouveau framework PHP que je ne connaissais pas, j'ai aussi appris à utiliser une librairie JavaScript et à me documenter dessus, le tout en autonomie.

Ce stage m'a aussi permis de conforter mon choix de continuer mes études en alternance pour gagner en expérience et en connaissances.

Je tiens à remercier Antoine de m'avoir permis de réaliser mon stage à Fleuron d'Anjou, Yohann de m'avoir accompagné tout le long de mon stage pour la réalisation de mon travail ainsi que toutes les personnes que j'ai pu rencontrer dans l'entreprise pour leur accueil et le partage de leurs connaissances.