

DOSSIER PROFESSIONNEL (DP)

Nom de naissance ▶ DELANNAY
Nom d'usage ▶ DELANNAY
Prénom ▶ Léa
Adresse ▶ 5 square des Bosquets, Bat A appt 57, 33700 Mérignac

Titre professionnel visé

Développeur web et web mobile

MODALITE D'ACCES :

- ☒ Parcours de formation
- ☐ Validation des Acquis de l'Expérience (VAE)

Présentation du dossier

Le dossier professionnel (DP) constitue un élément du système de validation du titre professionnel. **Ce titre est délivré par le Ministère chargé de l'emploi.**

Le DP appartient au candidat. Il le conserve, l'actualise durant son parcours et le présente **obligatoirement à chaque session d'examen.**

Pour rédiger le DP, le candidat peut être aidé par un formateur ou par un accompagnateur VAE.
Il est consulté par le jury au moment de la session d'examen.

Pour prendre sa décision, le jury dispose :

1. des résultats de la mise en situation professionnelle complétés, éventuellement, du questionnaire professionnel ou de l'entretien professionnel ou de l'entretien technique ou du questionnement à partir de productions.
2. du **Dossier Professionnel** (DP) dans lequel le candidat a consigné les preuves de sa pratique professionnelle
3. des résultats des évaluations passées en cours de formation lorsque le candidat évalué est issu d'un parcours de formation
4. de l'entretien final (dans le cadre de la session titre).

[Arrêté du 22 décembre 2015, relatif aux conditions de délivrance des titres professionnels du ministère chargé de l'Emploi]

Ce dossier comporte :

- ▶ pour chaque activité-type du titre visé, un à trois exemples de pratique professionnelle ;
- ▶ un tableau à renseigner si le candidat souhaite porter à la connaissance du jury la détention d'un titre, d'un diplôme, d'un certificat de qualification professionnelle (CQP) ou des attestations de formation ;
- ▶ une déclaration sur l'honneur à compléter et à signer ;
- ▶ des documents illustrant la pratique professionnelle du candidat (facultatif)
- ▶ des annexes, si nécessaire.

Pour compléter ce dossier, le candidat dispose d'un site web en accès libre sur le site.



<http://travail-emploi.gouv.fr/titres-professionnels>

Sommaire

Exemples de pratique professionnelle

Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité p. 5

- ▶ Réaliser une interface utilisateur web statique et responsive p. 5
- ▶ Développer une interface utilisateur web dynamique p. 16

Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité p. 27

- ▶ Créer une base de données p. 27
- ▶ Développer les composants d'accès aux données..... p. 32

Titres, diplômes, CQP, attestations de formation *(facultatif)* p. 39

Déclaration sur l'honneur p. 40

Documents illustrant la pratique professionnelle *(facultatif)* p. 41

Annexes *(Si le RC le prévoit)* p. 42

EXEMPLES DE PRATIQUE PROFESSIONNELLE

Activité-type 1

Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

Exemple n°1 ► Réaliser une interface utilisateur web statique et responsive

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

L'équipe Studio, que j'ai intégré durant 1.5 jour, travaille sur la partie maquettage graphique et maquettage html/css. Lors de mon séjour dans cette équipe, j'ai eu l'occasion de découper la maquette html/css d'une page de recherche d'offres particulière, concernant une opération entre Pôle Emploi et des partenaires.

Dans un premier temps, Jérémy Mahot m'a présenté les outils utilisés, le processus de création d'une maquette, et l'architecture des projets. Les graphistes réalisent des maquettes graphiques grâce aux outils Sketch et Invision. Ensuite, les intégrateurs html/css découpent ces maquettes graphiques pour réaliser les maquettes html/css.

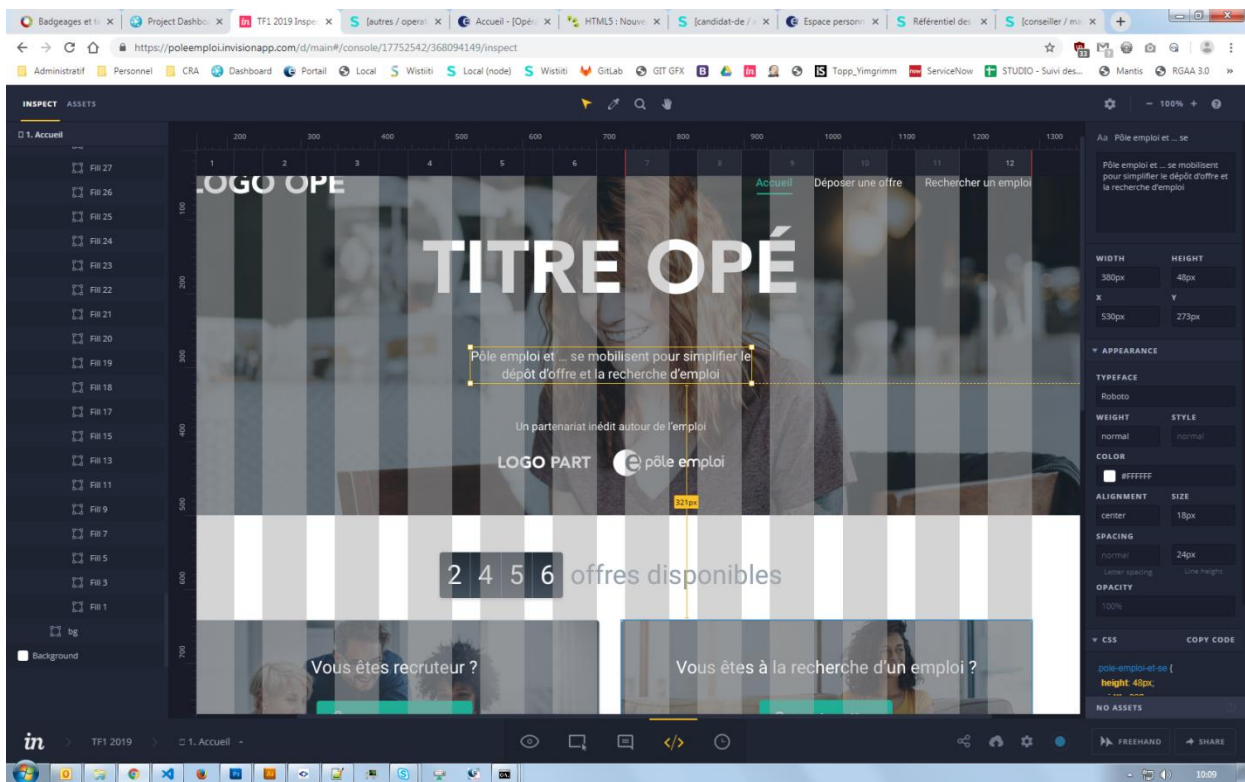
Nous utilisons le framework css/javascript Bootstrap, surchargé par des éléments graphiques propres à l'entreprise.

Voici la maquette à découper :



DOSSIER PROFESSIONNEL (DP)

L'outil Invision permet d'afficher la maquette graphique. Il indique les tailles, les couleurs, les marges entre tous les éléments composant la page.



Grâce à cela, j'ai pu dans un premier temps découper toute la page en blocs.

J'ai saisi ces blocs sous la forme de div dans mon fichier html afin d'avoir la structure de la page.

Jérémy ayant déjà fait d'autres pages concernant ce projet, le header et le footer ont été créés par ses soins sous forme de composants réutilisables, que j'ai intégré directement dans la page d'accueil sans avoir besoin de les créer.

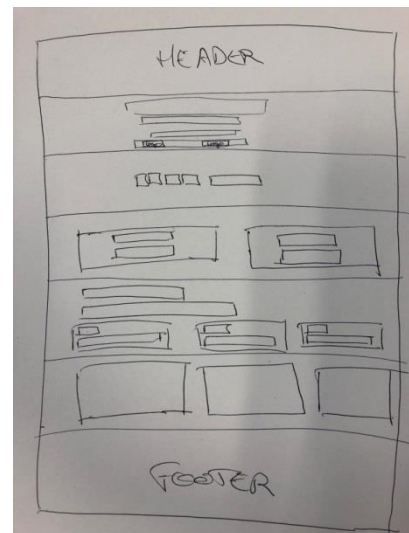
J'ai dégagé les éléments qui ne sont pas des div tels que les « span », les images « img », les liens « a », les titres « h1 », « h2 », « h3 »...

J'ai placé les classes Bootstrap permettant de placer les éléments au bon endroit : « container », « row », « col ». Ainsi, les éléments se positionnent correctement sur la page.

J'ai placé les classes définies par l'entreprise permettant de déclarer des tailles de titres de t1 à t5.

J'ai donné une classe à chacun de mes blocs principaux afin de pouvoir les cibler plus facilement par la suite avec css.

J'ai ajouté les logos partenaire et Pôle Emploi aux endroits dédiés sur les balises « img » dans le html grâce à l'attribut src.



J'ai redimensionné les images à intégrer à la page afin de pouvoir les utiliser, car elles étaient trop grandes par rapport aux conteneurs. Le but étant que la maquette soit la plus légère possible, il vaut mieux redimensionner les images avant de les intégrer. Les dimensions prises en compte sont celles que nous indique l'outil Invision.

```
<div class="accueil">
  <div class="heading">
    <div class="heading-content">
      <h1 class="heading-title">Lorem ipsum dolor sit amet, consectetur</h1>
      <div class="subtitle">Pôle emploi et ... se mobilisent pour simplifier le dépôt d'offre et la recherche d'emploi</div>
      <div>Un partenariat inédit autour de l'emploi</div>
      <div class="logos">
        
        
      </div>
    </div>
  </div>
  <div class="container">
    <div class="counter">
      <a href="#" class="counter-link" title="Voir les 2456 offres disponibles">
        <div class="panels" aria-hidden="true">
          <span>2</span><span>4</span><span>5</span><span>6</span>
        </div>
        <div class="t1"><span class="sr-only">2456 </span>offres disponibles</div>
      </a>
    </div>
    <div class="row">
      <div class="col-sm-6">
        <div class="block-category">
          <h2 class="t3">Vous êtes recruteur&nbsp;?</h2>
          <a href="#" class="btn btn-lg btn-primary" title="Déposer une offre dans le cadre de l'opération.">
            <span class="icon-op-ico-work" aria-hidden="true"></span>Déposer une offre
          </a>
        </div>
      </div>
      <div class="col-sm-6">
        <div class="block-category">
          <h2 class="t3">Vous êtes à la recherche d'un emploi&nbsp;?</h2>
          <a href="#" class="btn btn-lg btn-primary" title="Voir toutes les offres d'emploi.">
            <span class="icon-op-ico-search" aria-hidden="true"></span>Voir les offres
          </a>
        </div>
      </div>
    </div>
    <div class="block-edito">
      <h2 class="t1">Une opération spéciale</h2>
      <div class="subtitle">Pôle emploi et ... se mobilisent pour simplifier le dépôt d'offre et la recherche d'emploi</div>
      <div class="row">
        <div class="col-sm-4">
          <h3 class="t4">Lorem</h3>
          <div>Nunc molestie euismod sed orci nisi pulvinar eu sagittis vitae</div>
        </div>
        <div class="col-sm-4">
          <h3 class="t4">Ipsum</h3>
          <div>Egestas id turpis quisque ex metus pretium sed aliquet ut </div>
        </div>
        <div class="col-sm-4">
          <h3 class="t4">Dolores</h3>
          <div>Vivamus justo sapien sollicitudin eget viverra in hendrerit vitae justo</div>
        </div>
      </div>
    </div>
    <div class="block-communication">
      <div class="row">
        <div class="col-sm-4"><div class="communication">Publicité</div></div>
        <div class="col-sm-4"><div class="communication">Publicité</div></div>
        <div class="col-sm-4"><div class="communication">Publicité</div></div>
      </div>
    </div>
  </div>
</div>
```

J'ai ensuite créé le fichier `_operation-partenaire-accueil.scss`. J'ai ajouté ce fichier au fichier `operation-partenaire.scss`, qui contient la liste de tous les fichiers scss à utiliser pour les pages concernant

l'opération partenaire. Tout est découpé en sous fichiers qui sont eux-mêmes appelés par de plus gros fichiers.

Dans le fichier `_operation-partenaire-accueil.scss`, j'ai travaillé par blocs, c'est-à-dire que j'ai fait chacun des blocs principaux (cf ci-dessus) dégagés durant la découpe, sur lesquels j'ai appliqué les classes « heading », « counter », « block-category », « block-edito », et « block-communication ».

J'ai commencé par mettre en forme le header. Sa particularité est que l'en-tête, déjà intégré à la page, ne fait pas la même taille que l'en-tête de la maquette graphique. Cependant, il est imposé de travailler directement avec le composant déjà créé, et de surcharger ses classes dans mon fichier scss afin de rendre cet en-tête conforme à la maquette graphique.

```
.accueil{
  .heading{
    height: 550px;
    color: #fff;
    text-align: center;
    background: url(../img/accueil.jpg) center center;
    background-size: cover;
    background-repeat: no-repeat;
    padding-top: 125px;
    position: relative;
    animation: fadein 0.5s;

    &:before{
      content: '';
      display: block;
      position: absolute;
      left: 0;
      right: 0;
      top: 0;
      bottom: 0;
      background: #000;
      opacity: 0.6;
      z-index: 1;
    }

    .subtitle{
      max-width: 500px;
      margin: 0 auto 50px;
    }

    .logos{
      margin-top: 15px;
    }
  }
}
```

police en rem.

J'ai ensuite fait la deuxième partie contenant des images, c'est-à-dire le bloc de classe « block-category ».

```
.page-accueil{
  .header{
    box-shadow: none;
    background: none;
    position: relative;

    &.reduced{
      height: 100px;

      .media-left, .media-right{
        height: 100px;
      }
    }

    .media-right a:not(.active){
      color: #fff;
      transition: all 0.2s;
    }
  }

  .main{
    padding-top: 0;
    margin-top: -100px;
  }
}
```

J'ai donc appliqué des classes à la div de classe « heading » afin de faire en sorte que :

- la hauteur convienne,
- l'ombre et le fond disparaissent,
- la taille et les éléments de la navbar ne bouge plus,
- les titres et textes soient de la bonne taille,
- l'image soit placée en dessous et centrée,
- les marges internes et externes soient conformes à la maquette graphique,
- un voile opaque apparaisse exactement par-dessus l'image (placé entre l'image et la div de classe « heading-content »),
- tout ce qui est contenu dans la div de classe « heading-content » (titres, sous titres et logos) soit placé par-dessus ce voile opaque afin que les éléments restent de la couleur souhaitée.

Pôle emploi a également défini des mixins. Il s'agit de morceaux de code paramétrables et réutilisables n'importe où dans le fichier SASS. Le mixin utilisé dans ma page permet de déclarer des tailles de polices en px, et le mixin transforme cette taille de

Là encore, j'ai fait en sorte que les images soient placées en dessous, qu'un voile opaque apparaisse exactement par-dessus l'image (placé entre l'image et la div de classe « heading-content ») et que tout ce qui est contenu dans la div de classe « heading-content » (titres, sous titres et logos) soit placé par-dessus ce voile opaque afin que les éléments restent de la bonne couleur.

La particularité était de réussir à sélectionner le deuxième bloc d'image et de lui appliquer une image différente de celle contenue dans le premier bloc. Pour cela, j'ai utilisé les sélecteurs css « .col-sm-6 + .col_sm_6 .block-category », que j'ai surchargé avec une nouvelle image en fond.

```
.heading-content{
  z-index: 2;
  position: relative;
  max-width: 1040px;
  margin: 0 auto;
  padding: 0 20px;
}

.heading-title{
  margin-bottom: 50px;
  @include to-rem(58px);
  font-weight: bold;
}

.subtitle{
  @include to-rem(18px);
}

.col-sm-6 + .col-sm-6 .block-category{
  background: url(../img/accueil-offres.jpg) center center no-repeat;
  animation-delay: 0.7s;
}

.block-category{
  padding: 50px;
  text-align: center;
  background: url(../img/accueil-recruteur.jpg) center center no-repeat;
  background-size: cover;
  background-repeat: no-repeat;
  position: relative;
  border-radius: 6px;
  overflow: hidden;
  animation: fadein 0.5s forwards;
  animation-delay: 0.4s;
  opacity: 0;
  box-shadow: 0 2px 4px 0 rgba(0,0,0,.2);

  &:before{
    content: '';
    display: block;
    position: absolute;
    left: 0;
    right: 0;
    top: 0;
    bottom: 0;
    background: #000;
    opacity: 0.6;
    z-index: 1;
  }

  .t3{
    z-index: 2;
    margin-bottom: 30px;
    color: #fff;
    position: relative;
  }

  .btn{
    z-index: 2;
    position: relative;
  }
}
```

Concernant le bloc de classe « counter », la particularité est que le fond derrière les chiffres est dégradé, et que le bloc doit être cliquable. Dans une démarche d'accessibilité, j'ai ajouté des balises de classe « sr-only » et « aria-hidden » dans le html, afin que les lecteurs d'écrans interprètent correctement cette partie.

J'ai modifié les tailles et couleurs du texte, la couleur de fond, les bordures des span (sur lesquels j'ai appliqué la classe « inline-block » afin d'ajouter ce comportement qui n'est pas celui par défaut pour les span). J'ai modifié le bloc complet pour qu'il soit sous la forme d'un lien cliquable.

```
.panels{
  display: inline-block;
  background: linear-gradient(■rgb(34, 45, 56), ■rgb(67, 79, 91));
  border-radius: 4px;
}

.counter{
  text-align: center;
  animation: fadein 0.5s;
}

.counter-link {
  display: inline-block;
  margin: 30px 0 15px;
  padding: 15px;

  span{
    display: inline-block;
    color: □#fff;
    background-color: transparent;
    @include to-rem(36px);
    padding: 6px 10px;
    border-right: 1px solid ■#5e666f;
  }

  span:last-child{
    border-right: none;
  }

  .t1{
    display: inline-block;
    margin-left: 15px;
    color: ■#92A0AD;
    font-weight: 400;
  }
}
```

J'ai enfin mis en forme les blocs de classe « block-edito », et « block-communication ».

J'ai modifié les couleurs des textes, l'alignement du texte, la taille des polices, ajouté des marges, et modifié la couleur de fond des encadrés publicité.

```
.block-edito{
  margin-top: 50px;
  padding: 30px;
  background-color: #F8FAFC;
  border-radius: 8px;
  animation: fadein 0.5s;

  .t1{
    color: #222D38;
    font-weight: 400;
  }

  .subtitle{
    color: #92A0AD;
    margin: 15px 0 30px;
    @include to-rem(18px);
  }

  .t4{
    color: #1FAE96;
    @include to-rem(18px);
    font-weight: bold;
    padding-bottom: 10px;
  }

  .col-sm-4 div:nth-child(2){
    color: #222D38;
    @include to-rem(15px);
  }
}

.block-communication{
  margin-top: 50px;
  text-align: center;
  margin-bottom: 50px;
  animation: fadein 0.5s;

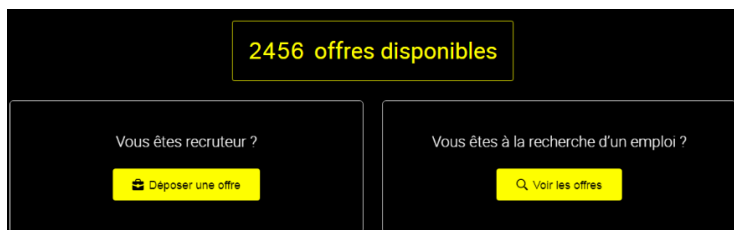
  .col-sm-4{
    .communication{
      background-color: #222d38;
      border-radius: 4px;
      padding: 85px 0;
      color: #fff;
      @include to-rem(24px);
    }
  }
}
```

Suite à cela, j'ai géré le mode contrasté proposé dans le pied de page. Il s'agit d'un mode plus épuré et plus lisible pour les personnes qui ont du mal à voir. Cela fait partie des mesures à mettre en place pour l'accessibilité. C'est une version alternative des contrastes qui permet d'avoir un contraste de plus de 7:1 entre la couleur de fond et le texte. Ce ratio est le ratio cible dans le référentiel d'accessibilité pour les administrations (RGAA). En dehors de cette version contrastée, on s'assure que les ratios dans la page sont au minimum de 3:1.

Pour cela, j'ai ajouté des éléments dans le fichier `_operation-partenaire.scss` existant. Le fichier `main`

principal importe tous les fichiers scss, ce qui fait que la page d'accueil peut être impactée par tous les fichiers scss. Je n'ai donc eu qu'à surcharger les classes, en ajoutant dans l'encart dédié au mode contrasté du fichier `_operation-partenaire.scss`, les noms des classes présentes dans mon html.

J'ai modifié les zones contenant des images, que j'ai enlevées pour mettre un fond noir à la place. J'ai ajouté une bordure blanche, enlevé la bordure du bloc de classe « header », et modifié la partie en gris pour qu'elle ait un fond noir, une bordure blanche, des textes blancs. J'ai mis les liens en jaune, et j'ai changé le fond dégradé des chiffres en noir.



Vous trouverez le rendu complet de la version contrastée en annexe.

```
.accueil{
  .heading{
    background: none;
    border-bottom: 1px solid #fff;
  }

  .col-sm-6 + .col-sm-6 .block-category{
    background: none;
    border: 1px solid #fff;
  }

  .block-category{
    background: none;
    border: 1px solid #fff;
  }

  .counter-link{
    border: 1px solid yellow;
    border-radius: 4px;
    padding: 15px 25px;
    margin-bottom: 30px;
  }

  .t1{
    color: yellow!important;
  }

  span{
    padding: 6px 1px;
    border-right: none;
    color: yellow!important;
  }
}

.panels{
  background: none;
}

.block-edito{
  background-color: #000;
  color: #fff;
  border: 1px solid #fff;

  .subtitle{
    color: #fff;
  }

  .col-sm-4 div:nth-child(2){
    color: #fff;
  }
}

.block-communication{
  .communication{
    border: 1px solid #fff;
  }
}
```

Ensuite, j'ai vérifié que la page est bien responsive, c'est-à-dire qu'elle s'adapte à toutes les tailles d'écran. J'ai ajouté les classes nécessaires dans un fichier existant nommé `_operation-partenaire-devices.scss`. Pour cela, j'ai vérifié dans le navigateur que le site était cohérent/joli à toutes les tailles d'écrans.

Pour les petites tailles d'écrans, j'ai dû adapter les marges et la taille de la police qui paraissait trop grande.

```
@media only screen and (max-width: 767px){
  .depot{...
  }
  .accueil{
    .col-sm-6 + .col-sm-6 .block-category, .block-edito,.block-edito .col-sm-4, .b
    margin-top: 20px;
  }
  .block-edito .col-sm-4:first-child, .block-communication .col-sm-4:first-child
  margin-top: 0;
  }
  .block-category{
    min-height: auto;
  }
  .t3{
    min-height: auto;
  }
}
```

A 1080px de largeur d'écran, il y avait un problème car les deux images du bloc de classe block-category n'étaient pas de la même taille ni alignées. J'ai donc ajouté une media query pour modifier cela.

```
@media only screen and (max-width: 1080px){
  .block-category{
    min-height: 260px;
    background-size: cover!important;
  }
  .t3{
    min-height: 78px;
  }
}
```

```
@media only screen and (max-width: 640px){
  .header-icon{ ...
  }

  .header-text{ ...
  }

  .header{ ...
  }

  .depot{ ...
  }

  .accueil{
    .heading-title{
      @include to-rem(30);
    }

    .block-category{
      padding: 50px 30px;
    }

    .block-edito .t1{
      @include to-rem(24);
    }

    .counter-link{
      padding: 0;
      margin: 30px 0;
    }

    .t1{
      @include to-rem(20);
      margin-left: 10px;
    }

    span{
      padding: 6px;
      @include to-rem(20);
    }
  }
}
```

Enfin, avec l'aide de Jérémy Mahot, j'ai réalisé ce que Pôle Emploi appelle un mini-audit :

- 1) Visuel : vérification que la page correspond bien à la maquette graphique et que tout est cohérent.
- 2) Html/css : l'outil plugin Chrome Web Developer permet de valider que le html est conforme aux règles du W3C, et que la page sans css est cohérente (les titres aux bons endroits etc)
- 3) Ux : vérification que l'enchaînement de la page est claire.
- 4) textes : l'outil Web Developer permet de visualiser la structure de la page, qui doit être cohérente. Vérification qu'il n'y a pas de fautes d'orthographe.
- 5) Compatibilité : vérification que la page apparait correctement sur les autres navigateurs (ici Firefox et Internet Explorer).
- 6) Accessibilité : désactivation du css pour vérifier que tous les boutons et liens ont des titres, puis vérifier que la version contrastée fonctionne correctement
- 7) Performances : dans l'onglet Network de Chrome, vérifier que la page n'est pas trop lourde. Ici, elle l'est, à cause des images. Cependant, les images qui seront affichées seront celles du partenaire.

Vous trouverez le rendu final de la page en annexe.

DOSSIER PROFESSIONNEL (DP)

2. Précisez les moyens utilisés :

Cette activité a été réalisée à l'aide de :

- IDE (Integrated Development Environment, environnement de développement) : Visual Studio Code
- Langages/framework : Nunjucks(html), SCSS, Bootstrap
- Matériels : ordinateur fixe, un écran, clavier, souris
- Système d'exploitation : Windows 7 professionnel
- Navigateur : Chrome, Firefox, Internet Explorer
- Outils : GitLab (gestion des projets et documentation), RTC (gestion des tâches), Wistiiti (gestion des maquettes existantes), Invision, Photoshop, Web Developer (plugin Chrome)

3. Avec qui avez-vous travaillé ?

J'ai travaillé avec Jérémy Mahot, intégrateur html/css.

Il m'a expliqué le fonctionnement de l'architecture des fichiers Nunjucks (moteur de template) et la logique des maquettes Pôle Emploi, afin que ma découpe soit cohérente avec le reste de l'application.

Il m'a également aidée dans le choix des noms de classe, toujours pour la cohérence dans les maquettes.

4. Contexte

Nom de l'entreprise, organisme ou association ► *DSI Pôle Emploi*

Chantier, atelier, service ► DSI, équipe Studio

Période d'exercice ► Du 11/06/2019 au 22/07/2019

5. Informations complémentaires (facultatif)

DOSSIER PROFESSIONNEL (DP)

Le code de l'application sur laquelle j'ai travaillé n'étant pas encore en production, il est classé confidentiel et ne peut pas être divulgué

Activité-type 1

Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

Exemple n°2 ▶ Développer une interface utilisateur web dynamique

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

L'équipe Opportunité Emploi que j'ai intégré développe la partie recherche d'offres et candidatures du site web pole-emploi.fr. Nous travaillons selon l'approche Agile avec la méthode Scrum.

Les user stories sont donc découpées en tâches, réalisées sous forme de sprint de 3 semaines.

Lors du premier sprint auquel j'ai participé, j'ai eu l'occasion d'intégrer la maquette html/css d'une page concernant un nouveau service que va proposer Pôle Emploi, et d'afficher du contenu dynamique sur une partie de la page intégrée.

1. Récupération du code du projet « dreusager » :

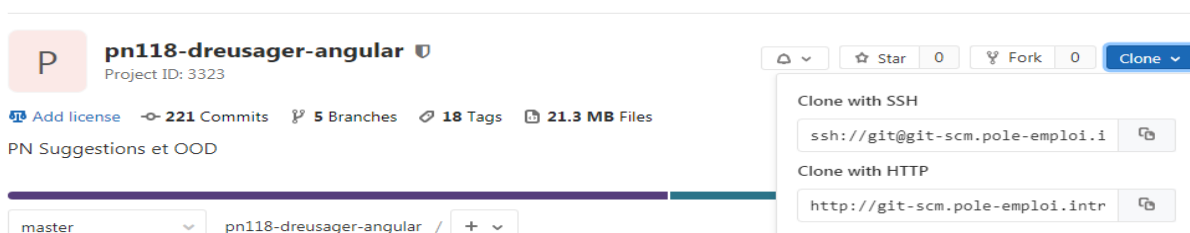
Depuis le repository GitLab dédiée au projet « dreusager », j'ai tiré le projet Angular, qui comportait déjà la page d'accueil, les pages d'indisponibilité/indisponibilité partielle et la page relative aux conseils personnalisés.

2. Intégration de la maquette :

Dans un premier temps, je suis allée sur l'outil de gestion des user stories afin de connaître précisément la tâche à effectuer et les règles de gestion associées.



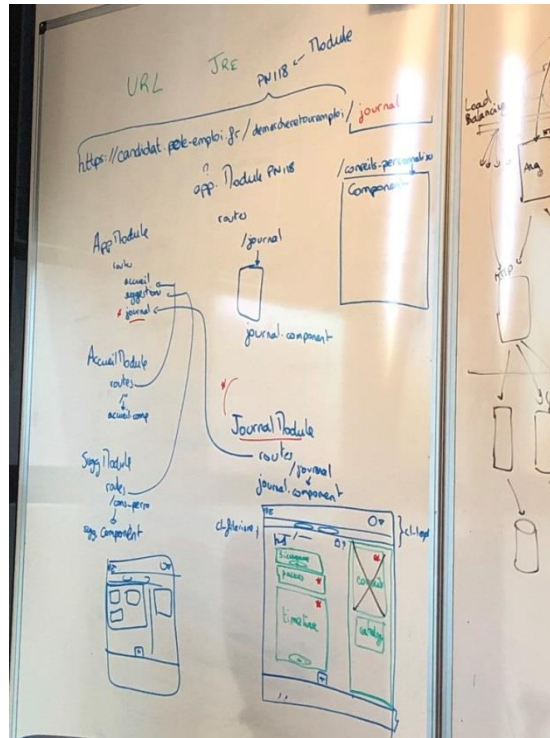
J'ai récupéré le code html/css depuis le repository GitLab dédié aux maquettes du projet.



Un vote a été soumis à toute l'équipe afin de déterminer quel sera le nom de la route (et donc de l'URL

finale) permettant d'accéder au journal de retour à l'emploi.

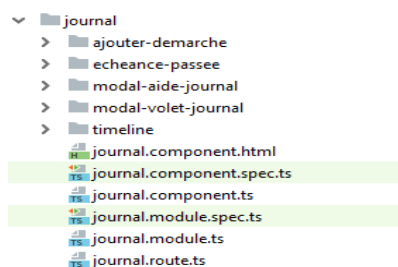
Avec l'aide de Benjamin Durand (lead technique de l'équipe), j'ai découpé au tableau la page à intégrer afin de dégager les composants Angular à créer. Les éléments de la page comportant l'appel à des données dynamiques ont été scindés en composants, afin qu'ils soient isolés et réutilisables.



J'ai créé le module `journal.module.ts` via la commande Angular CLI « `ng generate module journal` ».

Utiliser le CLI (command Line Interface) me permet de gagner du temps et d'éviter d'éventuelles erreurs de saisie lors de la création de ces éléments de l'application.

J'ai ensuite généré les composants via les commandes « `ng generate component journal` », « `ng generate component journal/echeance-passee` », « `ng generate component journal/modal-aide-journal` », « `ng generate component journal/modal-volet-journal` », et « `ng generate component journal/timeline` ».



Voici l'architecture créée par ces commandes :

J'ai ensuite créé un fichier `css` dans `app>assets>pole-emploi-framework>css>gfx` nommé `journal-recherche-emploi.css`, et j'y ai intégré le code `css` de la maquette. C'est dans ce dossier que sont rangés tous les fichiers `css` dédiés au projet « dreusager ».

J'ai téléchargé les icônes et images nécessaires à la page et je les ai mises dans le dossier correspondant dans `src/app/assets/css/gfx`.

J'ai créé la route permettant d'accéder à la page en ajoutant dans l'URL : /journal :

Création d'un fichier journal.routes.ts dans app/journal/ :

src/app/journal/journal.routes.ts 0 — 100644

```
1 + import {Routes} from '@angular/router';
2 + import {JournalComponent} from './journal.component';
3 +
4 + export const journalRoutes: Routes = [{path: 'journal', component: JournalComponent}];
```

Ajout de la route dans le fichier app.routing.module.ts :

```
const routes: Routes = [
  {
    path: '',
    component: AppLayoutComponent,
    canActivate: [AuthenticationGuard],
    children: [
      {
        path: '',
        component: MainComponent,
        children: [...accueilRoutes, ...suggestionsRoutes, ...journalRoutes]
      }
    ]
  }
]
```

Concernant l'intégration du code html, j'ai dans un premier temps intégré l'intégralité du code dans le fichier journal.component.html afin de vérifier que tout fonctionnait correctement.

J'ai ensuite scindé et intégré le code html de la maquette dans les composants Angular echeance-depassee, modal-aide, modal-volet, et timeline.

J'ai ajouté les composants echeance-depassee et timeline, ainsi que l'appel à la modal-aide, au composant journal :

```
<div class="journal">
  <cl-fil-ariane [linkItems]="navigationLinks" [openModal]="openModalAide.bind(this)"></cl-fil-ariane>
<app-echeance-passee></app-echeance-passee>

<app-timeline></app-timeline>
```

Et j'ai géré l'ouverture de la modale d'aide par une méthode dans journal.component.ts :

```
public openModalAide() {
  this.modalAide = this.modalService.show(ModalAideJournalComponent, {
    ...configModal
  });
  this.modalAide.content.getFocusOnFermer();
  const dataTagClic: DataTag = {
    chapters: ['JRE'],
    name: 'Popin_Aide',
    type: 'navigation'
  };
  this.tmsService.tag(dataTagClic);
  const dataTagPage: DataTag = { ...
  };
  this.tmsService.tag(dataTagPage);
}
```

J'ai intégré la modal-volet (qui permet d'ouvrir le détail d'une démarche) au composant timeline :

```
<a class="block-action-card"
  data-target="#PopinAction" data-toggle="modal"
  (click)="openModalVoletJournal()"
  role="button"
  title="Voir le détail de '{{demarche.libelleLong}}'">
```

Et j'ai géré l'ouverture de cette modale par une méthode dans `timeline.component.ts` :

```
openModalVoletJournal() {
  this.modalVoletJournal = this.modalService.show(ModalVoletJournalComponent,
  { ...configModal });
  this.modalVoletJournal.content.setVisible(true);
  this.renderer.addClass(document.body, 'modal-details-open');
  document.getElementsByClassName('modal')[1].setAttribute('class', 'modal modal-details
modal-conseil modal-action');
}
```

Durant cette intégration, j'ai rencontré quelques problèmes de conflits entre les fichiers/classes css de la page à intégrer et du projet global. Certaines classes de la page portaient le même nom que des classes du projet, ce qui surchargeait les classes et engendrait des écarts. J'ai donc listé l'intégralité des problèmes rencontrée, et cherché leur source. J'ai remonté les problèmes à Jérémie Mahot, l'intégrateur html css qui a produit la maquette, afin qu'il puisse modifier les fichiers en fonction. Suite à cela, j'ai intégré ses modifications dans les fichiers css et html dédiés.

3. Affichage du contenu dynamique de la Timeline

Dans un premier temps, je suis allée sur l'outil de gestion des user stories afin de connaître précisément la tâche à effectuer et les règles de gestion associées.

Story 200848: DAC 472 - US_02.02.05 - pe.fr - Timeline - Afficher le contenu de la page principale du JRE - hors entretiens, bloc échéance dépas...

Lorsque je clique sur "télécharger"
alors un pdf sera à télécharger (us spécifique prévue dans les prochains sprints au besoin)

TIMELINE

2. RG contenu de l'affichage

En tant que DE ayant réalisé des démarches et/ou ayant prévu des démarches
Lorsque je consulte la timeline
Alors je vois les informations suivantes sur la démarche en vue liste :

- libellé
- thème du référentiel
- convenu avec mon conseiller (lorsque c'est le cas)
- échéance dépassée (lorsque c'est le cas)
- statut
- l'info "aucun métier" n'est pas visible en si3 --> ça sera de la SI4

En tant que DE ayant réalisé un entretien avec mon conseiller
Lorsque je consulte la timeline
Alors je vois le libellé suivant pour mon entretien : « Vous avez eu un [type d'entretien] avec votre conseiller ».
--> reporté dans l'us 212208

DOSSIER PROFESSIONNEL (DP)

RG pour faire apparaître la question "l'avez-vous fait ?"

Seules les démarches dont l'origine est JRE DE ou JRE CONSEILLER ou PROJET D ACTION CONSEILLER peuvent avoir un statut autre que réalisé

Démarches dans le futur

En tant que DE ayant une démarche dans le futur, d'origine JRE DE ou JRE CONSEILLER
Alors je vois uniquement "oui" comme bouton cliquable pour la question "l'avez-vous fait ?"

En tant que DE ayant une démarche dans le futur, d'origine PROJET D ACTION CONSEILLER
Alors je ne vois pas la phrase "l'avez-vous fait ? ..." mais la phrase "Cette démarche est non modifiable"

Démarches à aujourd'hui

En tant que DE ayant une démarche à aujourd'hui, d'origine JRE DE
Alors je vois les boutons "oui" et "non" comme bouton cliquable pour la question "l'avez-vous fait ?"

En tant que DE ayant une démarche à aujourd'hui, d'origine JRE CONSEILLER
Alors je vois le bouton "oui" comme bouton cliquable pour la question "l'avez-vous fait ?"

En tant que DE ayant une démarche à aujourd'hui, d'origine PROJET D ACTION CONSEILLER
Alors je ne vois pas la phrase "l'avez-vous fait ? ..." mais la phrase "Cette démarche est non modifiable"

Démarches dans le passé

En tant que DE ayant une démarche dans le passé avec une échéance dépassée, d'origine JRE DE
Alors je vois les boutons "oui" et "non" comme bouton cliquable pour la question "l'avez-vous fait ?"

En tant que DE ayant une démarche dans le passé avec une échéance dépassée, d'origine JRE CONSEILLER
Alors je vois le bouton "oui" comme bouton cliquable pour la question "l'avez-vous fait ?"

En tant que DE ayant une démarche dans le passé avec une échéance dépassée, d'origine PROJET D ACTION CONSEILLER
Alors je ne vois pas la phrase "l'avez-vous fait ? ..." mais la phrase "Cette démarche est non modifiable"

En tant que DE ayant une démarche que le conseiller a ajouté (provenance JRE conseiller et Projet d'action)
alors je vois le tag "convenu avec mon conseiller" sur la démarche

En tant que DE ayant une démarche créée dans le JRE (DE ou conseiller)
alors le bouton "crayon" de modification est cliquable (peu importe le statut de la démarche) --> pour toutes les autres démarches le bouton est grisé

En tant que DE ayant une démarche associée à une période positionnée à aujourd'hui
alors je vois l'information "en cours du JJ/MM au JJ/MM"

En tant que DE ayant une démarche associée à une échéance positionnée à aujourd'hui
alors je vois l'information "à faire avant le JJ/MM"

En tant que DE ayant une démarche réalisée
alors je vois l'information "fait" sur la démarche

En tant que DE ayant une démarche annulée
alors je vois l'information "annulé" sur la démarche

Ensuite, avec l'aide de Benjamin Durand, nous avons mis en place l'appel au service permettant de récupérer les données depuis l'EX (façade d'exposition des données) concerné. Le back est géré par une autre équipe Pôle Emploi située à Aix-en-provence.

L'appel au service est fait sous la forme d'un observable auquel on s'abonne dans le composant dédié.

Dans le fichier `demarche.service.ts`, je déclare l'observable qui va chercher les données :

```
getTimeline(): Observable<Timeline> {  
  return this.http  
    .get<IDemarche[]>(this.appConfiguration.rest.ex077.timeline)  
    .pipe(map(demarches => new Timeline(demarches)));  
  // return of(new Timeline(this.bouchonDemarches));  
}
```

Dans le fichier `timeline.component.ts`, je m'abonne au retour du service :

```
ngOnInit() {  
  this.demarcheService.getTimeline().subscribe(  
    (timeline: Timeline) => {  
      this.changementEnCours = false;  
      this.timeline = timeline;  
    },  
    (erreur: Error) => {  
      this.hasErreur = true;  
      this.changementEnCours = false;  
      throw new ErreurMineur('Impossible de charger les démarches de la timeline : ' + erreur.message);  
    }  
  );  
};  
}
```

Grâce à cela, j'ai pu accéder aux données et mettre en place la gestion des données dynamiques dans la Timeline.

Les démarches sont typées grâce à l'interface IDemarche, interface créée au préalable par benjamin Durand, dans le but de fixer la structure de l'objet.

J'ai d'abord ajouté des directives structurales Angular (*ngIf) contenant toutes les conditions nécessaires à l'affichage des éléments dynamiques. Cependant, cela alourdissait le code html qui était devenu illisible.

Dans une démarche de « refactoring », j'ai donc créé des méthodes et j'y ai ajouté le contenu des directives structurales Angular (*ngIf) afin de gérer les données de manière plus lisible et plus maintenable. Ainsi, si une condition était amenée à évoluer ou à changer, il n'y aura besoin de la changer que dans la méthode dédiée et non pas partout dans le code html.

Voici les méthodes permettant de connaître la date du jour et son échéance, dans le fichier demarche-type.ts :

```
export class Timeline {
  jours: JourTimeline[] = [];
  constructor(demarches: IDemarche[]) {
    //Extrait les jours
    let datesDemarches: number[] = [];
    let demarcheAujourdHui = false;
    for (const demarche of demarches) {
      const dateDemarche: Date = new Date(demarche.dateTimeline);
      const indexDate = datesDemarches.indexOf(dateDemarche.getTime());
      if (indexDate < 0) {
        datesDemarches.push(dateDemarche.getTime());
        const jourTimeline: JourTimeline = new JourTimeline(dateDemarche);
        jourTimeline.addDemarche(demarche);
        this.jours.push(jourTimeline);
        if (jourTimeline.isAujourdHui()) {
          demarcheAujourdHui = true; } } else {
          const jourTimeline: JourTimeline = this.jours[indexDate];
          jourTimeline.addDemarche(demarche); } }
    //contient aujourd'hui
    if (!demarcheAujourdHui) {
      let aujourd'hui = new Date();
      aujourd'hui.setHours(0, 0, 0, 0);
      const jourTimeline: JourTimeline = new JourTimeline(aujourd'hui);
      this.jours.push(jourTimeline);
    } } }
export class JourTimeline {
  date: Date;
  demarches: IDemarche[] = [];

  constructor(date: Date) {
    this.date = date; }

  addDemarche(demarche: IDemarche): void {
    this.demarches.push(demarche); }

  isFutur(): boolean {
    return JourTimeline.comparerJour(this.date, new Date()) > 0; }

  isAujourdHui(): boolean {
    if (JourTimeline.comparerJour(this.date, new Date()) === 0) {
      return true; } }

  hasEcheanceDepasse(): boolean {
```

```
let echanceDepassee = false;
for (const demarche of this.demarches) {
  echanceDepassee = echanceDepassee || demarche.echanceDepassee; }
return echanceDepassee; }

isPasse(): boolean {
  return JourTimeline.comparerJour(this.date, new Date()) < 0; }

private static comparerJour(date1: Date, date2: Date): number {
  if (date1 && date2) {
    date1.setHours(0, 0, 0, 0);
    date2.setHours(0, 0, 0, 0);
    return date1.getTime() - date2.getTime(); }
  return -1; }}
```

Méthode permettant de gérer l'ajout de classes css sur l'échéance en fonction des données concernant le type de démarche afin de dynamiser l'affichage:

```
getClassEchance(demarche: IDemarche): string {
  switch (demarche.pourquoi.code) {
    case 'P01':
      return 'competences';
    case 'P02':
      return 'formation';
    case 'P03':
      return 'candidatures';
    case 'P04':
      return 'entretiens';
    case 'P05':
      return 'entreprise';
    default:
      return '';
  }
}
```

Méthode permettant de gérer l'ajout d'une classe css en fonction des données concernant l'échéance dépassée ou non :

```
getClassEchanceDepassee(demarche: IDemarche): string {
  if (demarche.echanceDepassee) {
    return 'outdated';
  }
  return '';
}
```

Méthode permettant de gérer l'ajout de classes css sur la démarche en fonction des données concernant l'état de la démarche :

```
getClassEtatDemarche(demarche: IDemarche): string {
  switch (demarche.etat.code) {
    case 'EC':
      return 'todo';
    case 'RE':
      return 'done';
    case 'AN':
      return 'cancelled';
    default:
      return '';
  }
}
```

Méthodes permettant de l'affichage des boutons et/ou messages, en fonction des critères définis dans l'US:

Page 23

J'ai ensuite créé les tests unitaires permettant de vérifier que les méthodes couvrent bien tous les cas / toutes les conditions possibles définies dans l'US. La création de ces tests permet également, en cas de modification ultérieure du code, d'alerter les développeurs sur les impacts de leurs ajouts dans le code existant.

J'ai créé cela dans le fichier dédié `timeline.component.specs.ts`. J'y ai ajouté toutes les dépendances dont a besoin mon fichier `timeline.component.ts` pour fonctionner, car les tests unitaires en ont également besoin.

Etienne Martin m'a expliqué comment on écrit des tests unitaires en Angular, puis j'ai écrit un test par méthode. Chaque test unitaire couvre le comportement si le cas fonctionne (s'il est vrai) et s'il ne fonctionne pas (s'il est faux). Concernant les méthodes qui ajoutent des classes css, je vérifie que le code ajoute bien la bonne classe au bon endroit.

```
describe('TimelineComponent', () => {
  let component: TimelineComponent;
  let fixture: ComponentFixture<TimelineComponent>;

  beforeEach(
    async(() => {
      TestBed.configureTestingModule({
        imports: [HttpClientTestingModule],
        declarations: [TimelineComponent, IndisponibilitePartielleComponent],
        providers: [
          {provide: AppConfigurationToken, useValue: environment},
          {provide: BsModalService, useClass: BsModalServiceMock},
          {provide: DemarcheService, useClass: DemarcheServiceMock}
        ]
      }).compileComponents();
    })
  );

  beforeEach(() => {
    fixture = TestBed.createComponent(TimelineComponent);
    component = fixture.componentInstance;
    registerLocaleData(localeFr, 'fr');
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });

  it('doit tester si la démarche est modifiable', function() {
    expect(component.isDemarcheModifiable(demarcheJREConseillerEnCours)).toBeTruthy();
    expect(component.isDemarcheModifiable(demarcheJREDEEnCours)).toBeTruthy();
    expect(component.isDemarcheModifiable(demarcheActionRealise)).toBeFalsy();
  });

  it('doit tester si le message non modifiable s\'affiche', function() {
    expect(component.isAffichageMessageNonModifiable(demarcheActionEnCours)).toBeTruthy();
    expect(component.isAffichageMessageNonModifiable(demarcheJREConseillerEnCours)).toBeFalsy();
    expect(component.isAffichageMessageNonModifiable(demarcheJREDEEnCours)).toBeFalsy();
    expect(component.isAffichageMessageNonModifiable(demarcheActionAnnule)).toBeFalsy();
    expect(component.isAffichageMessageNonModifiable(demarcheActionRealise)).toBeFalsy();
  });

  it('doit tester si l\'encadré contenant le bouton oui s\'affiche', function() {
    expect(component.isAffichageLavezVousFait(demarcheJREConseillerEnCours)).toBeTruthy();
  });
});
```


DOSSIER PROFESSIONNEL (DP)

2. Précisez les moyens utilisés :

Cette activité a été réalisée à l'aide de :

- IDE (Integrated Development Environment, environnement de développement) : IntelliJ
- Framework : Angular 6
- Matériels : ordinateur fixe, deux écrans, clavier, souris, tableau blanc, SCRUM Board
- Méthodes : Agiles (SCRUM)
- Système d'exploitation : Windows 7 professionnel
- Navigateurs : Chrome, Firefox
- Outils : GitLab (gestion des projets et documentation), RTC (gestion des user stories), Wistiiti (gestion des maquettes), Jenkins (intégration continue), Sonar (qualité du code)
- Communication : Skype Entreprise, Mattermost

3. Avec qui avez-vous travaillé ?

J'ai travaillé avec Benjamin Durand, développeur et lead technique dans l'équipe Opportunité Emploi à la DSI de Pôle Emploi. Il a fait le service permettant d'appeler le serveur pour récupérer les données et m'a aidée lorsque j'étais bloquée sur des problèmes techniques. Il a également relu mon code afin de m'aider à mieux le structurer.

Etienne Martin, développeur dans l'équipe Opportunité Emploi, a également travaillé sur une partie de la timeline présentée, concernant l'affichage du bouton « modifier » grisé. Il m'a aidée à mieux structurer mon code en méthodes réutilisables.

Jérémy Mahot, intégrateur html/css, a fourni la maquette html/css que j'ai dû intégrer. Il a également résolu des problèmes de conflits entre les fichiers/classes css.

4. Contexte

Nom de l'entreprise, organisme ou association ► **DSI Pôle Emploi**

Chantier, atelier, service ► Département Recherche D'Emploi (RDE), équipe Opportunité Emploi

Période d'exercice ► Du 11/06/2019 au 22/07/2019

5. Informations complémentaires (facultatif)

DOSSIER PROFESSIONNEL (DP)

Le code de l'application sur laquelle j'ai travaillé n'étant pas encore en production, il est classé confidentiel et ne peut pas être divulgué

Activité-type 2

Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité

Exemple n°1 ► Créer une base de données

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Dans le cadre de la formation développeur web et web mobile, j'ai réalisé une application personnelle en Angular 7 et NodeJs. Pour cela, j'ai dû établir le besoin et créer la base de données correspondante.

J'ai choisi d'utiliser une base de données relationnelle car les données sont amenées à être souvent mises à jour et possèdent des relations fortes entre elles.

Afin d'établir la base de données, j'ai dans un premier temps dégagé les objets et leurs caractéristiques en fonction du cahier des charges et des user stories. J'ai aussi établi les relations entre les objets.

Objets	Identifiant	Propriétés	Dans la table
vêtement	id_vet	nom, image, description, catégorie, marque, note, utilisateur, caractéristique, couleur, occasion, tenue	Clé étrangère
couleur	id_coul	libellé, couleur	Table associative associant un vêtement à plusieurs critères
marque	id_marque	nom	Table associative
occasion	id_occasion	libellé	
catégorie	id_cat	libellé	
note	id_note	numéro	
caractéristique	id_caract	libellé, caractéristique	
tenue	id_tenue	libellé	
user	id_user	pseudo, login, mdp	

J'ai ensuite établi le modèle conceptuel des données (MCD) sur le logiciel Power AMC, que vous trouverez en annexe. J'ai traduit ce MCD en modèle logique des données (MLD), sur le logiciel Power AMC. Vous le trouverez également en annexe.

J'ai ainsi pu générer le code SQL. Je l'ai retravaillé pour plus de lisibilité, notamment afin que les noms des clés étrangères ne soient pas les mêmes que ceux des clés primaires. Pour des questions de re jouabilité, j'ai ajouté en début de fichier les instructions SQL permettant de créer la base de données si cette dernière n'existe pas, et d'utiliser cette base de données. Le script SQL permet aussi de supprimer les tables si elles existent avant de les créer afin d'éviter d'éventuelles incohérences dans les données.

```
CREATE DATABASE IF NOT EXISTS dressing;
USE dressing;
drop table if exists CARACTERISTIQUE;
drop table if exists CARACT_ASSOC;
drop table if exists CATEGORIE;
drop table if exists COULEUR;
drop table if exists COUL_ASSOC;
drop table if exists MARQUE;
drop table if exists NOTE;
drop table if exists OCCASION;
drop table if exists TENUE;
drop table if exists USER;
drop table if exists VET_TEN_ASSOC;
drop table if exists VETEMENT;
drop table if exists VET_CARACT_ASSOC;
drop table if exists VET_COUL_ASSOC;
drop table if exists VET_OCCAS_ASSOC;
```

DOSSIER PROFESSIONNEL (DP)

J'ai intégré le code SQL dans l'onglet SQL de phpMyAdmin (outil open source disponible dans le navigateur qui est utilisé pour administrer MySQL), et j'ai ajouté un jeu de données dans chaque table afin de pouvoir tester l'application au fur et à mesure de son développement.



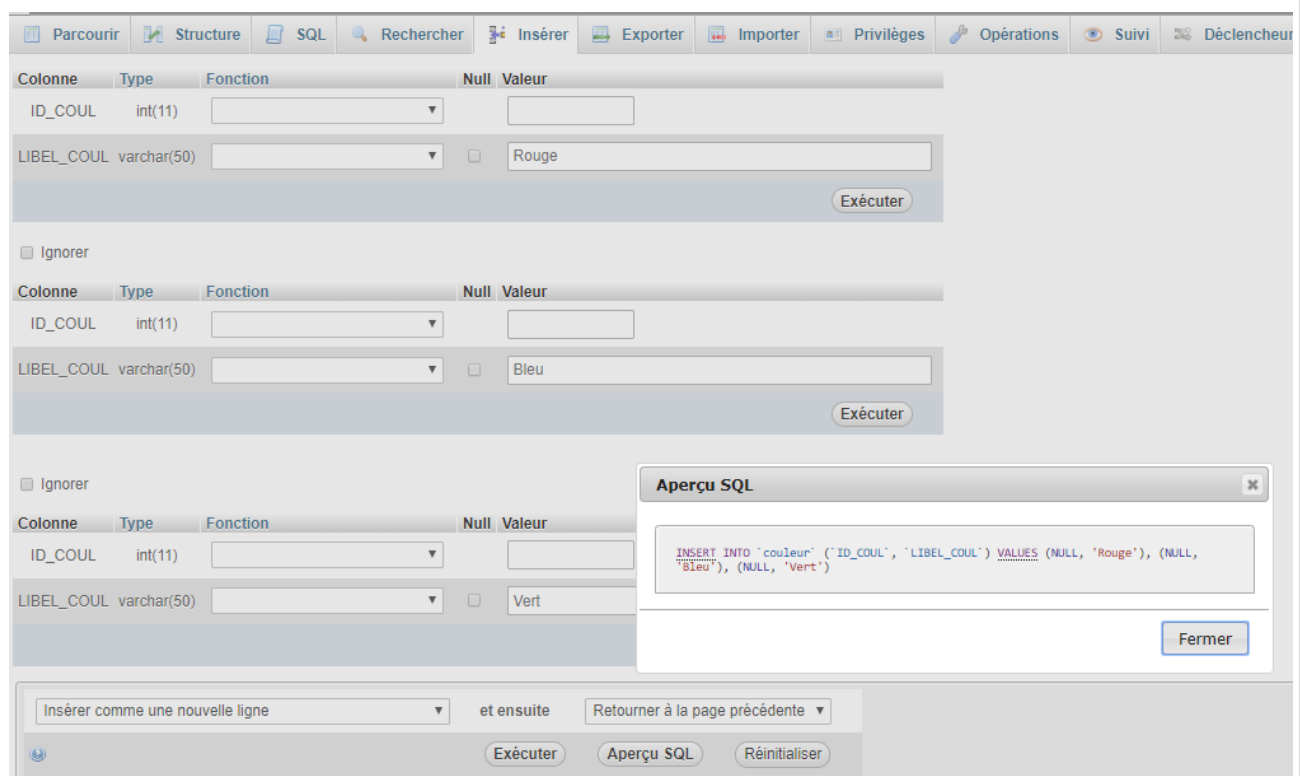
Seigneur: 127.0.0.1

Bases de données SQL État Comptes utilisateurs Exporter

Exécuter une ou des requêtes SQL sur le serveur « 127.0.0.1 »:

```
1 CREATE DATABASE IF NOT EXISTS dressing;
2 USE dressing;
3
4 drop table if exists CARACTERISTIQUE;
5 drop table if exists CARACT_ASSOC;
6 drop table if exists CATEGORIE;
7 drop table if exists COULEUR;
8 drop table if exists COUL_ASSOC;
9 drop table if exists MARQUE;
10 drop table if exists NOTE;
11 drop table if exists OCCASION;
12 drop table if exists TENUE;
13 drop table if exists USER;
14 drop table if exists VET_TEN_ASSOC;
15 drop table if exists VETEMENT;
16 drop table if exists VET_CARACT_ASSOC;
17 drop table if exists VET_COUL_ASSOC;
18 drop table if exists VET_OCCAS_ASSOC;
19
20 /*=====*/
21 /* Table : CARACTERISTIQUE */
22 /*=====*/
23 create table CARACTERISTIQUE
24 (
25     ID_CARACT          Int Auto_increment NOT NULL ,
26     LIBEL_CARACT       varchar(255),
```

Effacer Format Récupérer la requête auto-sauvegardée



Parcourir Structure SQL Rechercher Insérer Exporter Importer Privileges Opérations Suivi Déclencheur

Colonne	Type	Fonction	Null	Valeur
ID_COUL	int(11)			
LIBEL_COUL	varchar(50)		<input type="checkbox"/>	Rouge

Exécuter

☐ Ignorer

Colonne	Type	Fonction	Null	Valeur
ID_COUL	int(11)			
LIBEL_COUL	varchar(50)		<input type="checkbox"/>	Bleu

Exécuter

☐ Ignorer

Colonne	Type	Fonction	Null	Valeur
ID_COUL	int(11)			
LIBEL_COUL	varchar(50)		<input type="checkbox"/>	Vert

Exécuter

Aperçu SQL

```
INSERT INTO `couleur` (`ID_COUL`, `LIBEL_COUL`) VALUES (NULL, 'Rouge'), (NULL, 'Bleu'), (NULL, 'Vert')
```

Fermer

Insérer comme une nouvelle ligne et ensuite Retourner à la page précédente

Exécuter Aperçu SQL Réinitialiser

J'ai ensuite réalisé une sauvegarde de la base de données contenant tous les jeux de données. Une sauvegarde mensuelle est prévue.

Pour cela, je vais dans phpMyAdmin, dans la base de données « Dressing », sous l'onglet « Exporter », et j'exporte la base de données au format SQL.

DOSSIER PROFESSIONNEL (DP)

Exportation des tables depuis la base de données « dressing »

Modèles d'exportation :

Nouveau modèle : Modèles existants :

Nom du modèle Créer Modèle : -- Sélectionner un modèle -- Mettre à jour Supprimer

Méthode d'exportation :

☒ Rapide, n'afficher qu'un minimum d'options
☐ Personnalisée, afficher toutes les options possibles

Format :

SQL

Exécuter

J'ai travaillé sur ce projet depuis plusieurs ordinateurs. Pour restaurer ou simplement ajouter la base de données précédemment enregistrée, j'utilise également phpMyAdmin. Je vais sur la page d'accueil du site, dans l'onglet « Importer », je clique sur « Choisir un fichier », je vérifie que le jeu de caractères du fichier est bien utf-8 et que le format est bien sur SQL, puis j'exécute.

Importation dans le serveur courant

Fichier à importer :

Le fichier peut être compressé (gzip, bzip2, zip) ou non.
Le nom du fichier compressé doit se terminer par `.[format].[compression]`. Exemple : `.sql.zip`

Parcourir les fichiers : Choisir un fichier Aucun fichier choisi (Taille maximale : 2 048kio)

Il est également possible de glisser-déposer un fichier sur n'importe quelle page.

Jeu de caractères du fichier : utf-8

Importation partielle :

☒ Permettre l'interruption de l'importation si la limite de temps configurée dans PHP est sur le point d'être atteinte.

Ignorer ce nombre de requêtes (pour SQL), à partir du début : 0

Autres options :

☒ Activer la vérification des clés étrangères

Format :

SQL

Options spécifiques au format :

Mode de compatibilité SQL : NONE

☒ Ne pas utiliser AUTO_INCREMENT pour la valeur zéro

Exécuter

DOSSIER PROFESSIONNEL (DP)

2. Précisez les moyens utilisés :

Cette activité a été réalisée à l'aide de :

- IDE (Integrated Development Environment, environnement de développement) : Visual Studio Code
- Langage : SQL
- Matériels : ordinateur portable, deux écrans, clavier, souris
- Système d'exploitation : Windows 10
- Navigateur : Chrome
- Outils : GitHub (gestion des projets), PowerAMC, phpMyAdmin, Excel, Xampp(Apache, MySQL)

3. Avec qui avez-vous travaillé ?

J'ai travaillé seule sur ce projet personnel.

4. Contexte

Nom de l'entreprise, organisme ou association ► **IFPA**

Chantier, atelier, service ► **Projet MonDressing**

Période d'exercice ► Du **16/01/2019** au **22/07/2019**

5. Informations complémentaires (facultatif)

DOSSIER PROFESSIONNEL (DP)

Cliquez ici pour taper du texte.

Activité-type 2

Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité

Exemple n°2 ▶ Développer les composants d'accès aux données

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Dans le cadre de la formation développeur web et web mobile, j'ai réalisé une application personnelle en Angular 7 et NodeJs. Pour cela, j'ai dû développer les composants d'accès aux données permettant au front d'envoyer et recevoir les données de la base de données.

J'ai utilisé le framework Express et le serveur NodeJs.

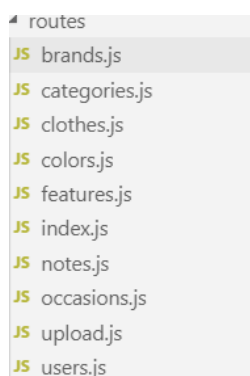
J'ai initialisé le projet grâce aux commandes « npm install -g express », « express » et « npm install ».

J'ai défini les routes sur lesquelles le front pourra venir chercher les ressources nécessaires, dans le fichier app.js.

```
var brandsRouter = require('./routes/brands');
var categoriesRouter = require('./routes/categories');
var clothesRouter = require('./routes/clothes');
var colorsRouter = require('./routes/colors');
var featuresRouter = require('./routes/features');
var notesRouter = require('./routes/notes');
var occasionsRouter = require('./routes/occasions');
var uploadRouter = require('./routes/upload.js'); //upload images
var usersRouter = require('./routes/users');

.use('/api/brands', brandsRouter);
.use('/api/categories', categoriesRouter);
.use('/api/clothes', clothesRouter);
.use('/api/colors', colorsRouter);
.use('/api/features', featuresRouter);
.use('/api/notes', notesRouter);
.use('/api/occasions', occasionsRouter);
.use('/api/upload', uploadRouter); //upload
.use('/api/users', usersRouter);
```

J'ai créé les ressources dans le dossier routes.



```
routes
├── brands.js
├── categories.js
├── clothes.js
├── colors.js
├── features.js
├── index.js
├── notes.js
├── occasions.js
├── upload.js
└── users.js
```

J'ai créé un dossier dbaccess qui contient un fichier index.js. Ce fichier contient les accès à la base de données ainsi que toutes les fonctions permettant de requêter en base.

Je vais vous présenter la ressource « clothes », car il s'agit de la ressource la plus représentative en termes de fonctionnalités.

Dans la ressource clothe.js, je joue des fonctions qui récupèrent les requêtes transmises par le front et j'exécute une fonction située dans index.js qui va chercher ou intégrer les informations dans la base de

données.

Un retour est ensuite fait au front contenant soit un code http d'erreur, soit les informations demandées et un code http de succès.

Lorsque le front appelle en get (méthode http permettant la récupération d'une ressource) la route `/api/clothes/`, cela joue la fonction suivante :

```
//READ ALL CLOTHES
router.get('/', function (req, res, next) {
  console.log("----> call read all clothes");
  dbacc.readClothes(function (err, data) {
    if (err) {
      res.sendStatus(500);
      return;
    }
    if (data == 0 || data == null || data == undefined || data == "") {
      res.sendStatus(204); //no content
      return;
    }
    res.send(data);
  });
});
```

Qui elle-même appelle la fonction `readClothes` contenue dans `index.js` :

```
//LISTE DE TOUS LES VETEMENTS EN BASE DE DONNEES - ALL
module.exports.readClothes = function (fct) {
  connection.query('SELECT * FROM vetement ORDER BY NOM_VET ASC', (err, results) => {
    if (err) {
      console.error(err);
      fct(err, null);
      connection.end();
      return;
    }
    fct(null, results);
    console.log(results);
  });
}
```

La fonction `readClothes` exécute une requête SQL préparée qui va renvoyer toutes les informations concernant un vêtement, par ordre alphabétique. En cas d'erreur, le code d'erreur 500 « Internal Server Error » est renvoyé au front et la fonction s'arrête. S'il n'y a pas d'erreur, le contenu de la requête SQL renvoyé par la base est transmis accompagné d'un code 200 « OK ». Si le contenu est vide ou null ou undefined ou égal à 0, le code http 204 « No Content » est renvoyé au front. Sinon, les informations sont transmises au front.

Lorsque le front appelle en post (méthode http permettant la création d'une ressource) la route `/api/clothes/`, cela joue la fonction suivante :

```
router.post('/', function (req, res, next) {
  console.log("req.body ----> " + req.body);
  var contenuRecuReq = JSON.stringify(req.body);
  console.log("req.body json.stringify ----> " + contenuRecuReq);
  if (!req.body.NOM_VET) {
    res.sendStatus(400);
    return;
  }
  dbacc.createClothe(req.body, function (err, data) {
    if (err) {
      res.sendStatus(500);
      return;
    }
    res.send(data);
  })
});
```

Qui elle-même appelle la fonction `readClothes` contenue dans `index.js` :

```
//LISTE DE TOUS LES VETEMENTS EN BASE DE DONNEES - ALL
module.exports.readClothes = function (fct) {
  connection.query('SELECT * FROM vetement ORDER BY NOM_VET ASC', (err, results) => {
    if (err) {
      console.error(err);
      fct(err, null);
      connection.end();
      return;
    }
    fct(null, results);
    console.log(results);
  });
}
```

La fonction `readClothes` exécute une requête SQL qui va renvoyer toutes les informations concernant un vêtement, par ordre alphabétique. En cas d'erreur, le code d'erreur 500 « Internal Server Error » est renvoyé au front et la fonction s'arrête. S'il n'y a pas d'erreur, le contenu de la requête SQL renvoyé par la base est transmis. Si le contenu est vide ou null ou undefined ou égal à 0, le code http 204 « No Content » est renvoyé au front. Sinon, les informations sont transmises au front.

Lorsque le front appelle en post (méthode http permettant la création d'une ressource) la route `/api/clothes/`, cela joue la fonction suivante :

```
router.post('/', function (req, res, next) {
  console.log("req.body ----> " + req.body);
  var contenuRecuReq = JSON.stringify(req.body);
  console.log("req.body json.stringify ----> " + contenuRecuReq);
  if (!req.body.NOM_VET) {
    res.sendStatus(400);
    return;
  }
  dbacc.createClothe(req.body, function (err, data) {
    if (err) {
      res.sendStatus(500);
      return;
    }
    res.send(data);
  })
});
```

Qui elle-même appelle la fonction `createClothe` contenue dans `index.js` :

```
//CREATION D'UN VETEMENT EN BASE DE DONNEES
module.exports.createClothe = function (obj, fct) {
  var idVet;

  var sql1 = "INSERT INTO vetement (FK_ID_CAT, FK_ID_MARQUE, FK_ID_NOTE, FK_ID_USER, NOM_VET, IMG_VET, DESCRIPT_VET) VALUES(?, ?, ?, ?, ?, ?, ?)";
  var inserts1 = [obj.FK_ID_CAT, obj.FK_ID_MARQUE, obj.FK_ID_NOTE, obj.FK_ID_USER, obj.NOM_VET, obj.IMG_VET, obj.DESCRPT_VET];
  // création du vêtement en base de données
  connection.query(mysql.format(sql1, inserts1), (err, results) => {
    if (err) {
      console.error(err);
      fct(err, null);
      return;
    }
    //récupération de l'id du vêtement créé
    idVet = results.insertId;
    //création des associations
    var featureArray = obj.ID_CARACT;
    featureArray.forEach(function (item) {
      var sql2 = "INSERT INTO vet_caract_assoc (ID_VET, ID_CARACT) VALUES(" + idVet + ", ?)";
      var inserts2 = [item];
      console.log("CARACTERISTIQUE" + inserts2);
      connection.query(mysql.format(sql2, inserts2), (err) => {
        if (err) {
          console.error(err);
          fct(err, null);
          return;
        }
      });
    });
  });
  var colorArray = obj.ID_COUL;
  colorArray.forEach(function (item) {
    var sql3 = "INSERT INTO vet_coul_assoc (ID_VET, ID_COUL) VALUES(" + idVet + ", ?)";
    var inserts3 = [item];
    console.log("COULEUR" + inserts3);
    connection.query(mysql.format(sql3, inserts3), (err) => {
      if (err) {
        console.error(err);
        fct(err, null);
        return;
      }
    });
  });
  var occasArray = obj.ID_OCCAS;
  occasArray.forEach(function (item) {
    var sql4 = "INSERT INTO vet_occas_assoc (ID_VET, ID_OCCAS) VALUES(" + idVet + ", ?)";
    var inserts4 = [item];
    console.log("OCCASION" + inserts4);
    connection.query(mysql.format(sql4, inserts4), (err) => {
      if (err) {
        console.error(err);
        fct(err, null);
        return;
      }
    });
  });
  fct(null, results);
});
}
```

La fonction `createClothe` exécute une requête SQL qui va insérer en base les éléments de l'objet envoyé par le front. Les éléments sont insérés dans les diverses tables correspondantes : la table *vetement*, mais également les tables d'association *vet_caract_assoc*, *vet_coul_assoc*, et *vet_occas_assoc*.

En cas d'erreur, le code d'erreur 500 « Internal Server Error » est renvoyé au front et la fonction s'arrête. Si l'objet envoyé par le front ne contient pas au moins le nom du vêtement, le code http 400 « Bad Request » est renvoyé au front. S'il n'y a pas d'erreur, le code http 200 « OK » est transmis au front, et les quatre requêtes insertion sont jouées les unes à la suite des autres.

2. Précisez les moyens utilisés :

DOSSIER PROFESSIONNEL (DP)

Cette activité a été réalisée à l'aide de :

- IDE (Integrated Development Environment, environnement de développement) : Visual Studio Code
- Langages/framework : NodeJs(Javascript), Express
- Matériels : ordinateur portable, deux écrans, clavier, souris
- Système d'exploitation : Windows 10
- Navigateur : Chrome
- Outils : GitHub (gestion des projets), Postman

3. Avec qui avez-vous travaillé ?

J'ai travaillé seule sur ce projet personnel.

4. Contexte

Nom de l'entreprise, organisme ou association ► **IFPA**

Chantier, atelier, service ► **Projet MonDressing**

Période d'exercice ► Du **16/01/2019** au **22/07/2019**

5. Informations complémentaires (facultatif)

DOSSIER PROFESSIONNEL (DP)

Cliquez ici pour taper du texte.

DOSSIER PROFESSIONNEL (DP)

Titres, diplômes, CQP, attestations de formation

(facultatif)

Intitulé	Autorité ou organisme	Date
BTS Assistant(e) de Gestion PME PMI	EBBS Bordeaux	09/07/2014
Baccalauréat Sciences et technologies de la Gestion	L'assomption Sainte Clotilde	09/07/2010
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.

DOSSIER PROFESSIONNEL (DP)

Déclaration sur l'honneur

Je soussigné(e) [prénom et nom] *Léa DELANNAY*..... ,
déclare sur l'honneur que les renseignements fournis dans ce dossier sont exacts et que je
suis l'auteur(e) des réalisations jointes.

Fait à *Mérignac*..... le *29/07/2019*.....

pour faire valoir ce que de droit.

Signature :

DOSSIER PROFESSIONNEL (DP)

Documents illustrant la pratique professionnelle

(facultatif)

Intitulé
Capture d'écran de l'interface web statique grande taille
Capture d'écran de l'interface web statique petite taille
Capture d'écran de l'interface web statique en version contrastée

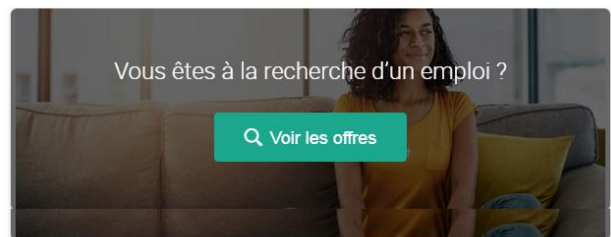
DOSSIER PROFESSIONNEL (DP)

ANNEXES

(Si le RC le prévoit)



2 4 5 6 offres disponibles



Une opération spéciale

Pôle emploi et ... se mobilisent pour simplifier le dépôt d'offre et la recherche d'emploi

Lorem

Nunc molestie euismod sed orci nisi pulvinar eu sagittis vitae

Ipsum

Egestas id turpis quisque ex metus pretium sed aliquet ut

Dolores

Vivamus justo sapien sollicitudin eget viverra in hendrerit vitae justo

Publicité

Publicité

Publicité



Capture d'écran de l'interface web statique petite taille

