

DOSSIER PROFESSIONNEL (DP)

Nom de naissance ▶ MECKLER
Nom d'usage ▶ MECKLER
Prénom ▶ Christophe
Adresse ▶ 4, rue Pasteur 54110 Dombasle sur Meurthe

Titre professionnel visé

Développeur Web/ Web mobile

MODALITÉ D'ACCÈS :

- ☒ 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.

DOSSIER PROFESSIONNEL (DP)



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

Sommaire

Exemples de pratique professionnelle

Intitulé de l'activité-type n° 1 - Développer la partie front-end d'une application web en intégrant les recommandations de sécurité	p.	6
▶ Intitulé de l'exemple n° 1 – Projet pour l'examen	p.	6
I. Présentation du projet	p.	6
II. Gestion du projet	p.	6
III. Maquetter une application	p.	7
IV. Réaliser une interface utilisateur web statique et adaptable	p.	9
V. Développer une interface utilisateur web dynamique	p.	13
VI. Réaliser une interface utilisateur avec une solution de gestion de contenu	p.	17
VII. La phase de test de la partie front-end	p.	17
VIII. Sécurité	p.	17
Intitulé de l'activité-type n° 2 - Développer la partie back-end d'une application web en intégrant les recommandations de sécurité	p.	19
▶ Intitulé de l'exemple n° 1 – Projet pour l'examen.....	p.	19
I. Établir les besoins	p.	19
II. Conception et création d'une base de données	p.	19
III. Développer les composants d'accès aux données	p.	20
IV. Développer la partie back-end d'une application web ou web mobile	p.	23
V. Mettre en œuvre des composants dans une application de gestion de contenu	p.	30
VI. La phase de test de la partie back-end	p.	30
VII. Sécurité	p.	31
Titres, diplômes, CQP, attestations de formation (facultatif)	p.	<input type="text"/>

DOSSIER PROFESSIONNEL (DP)

Déclaration sur l'honneur

p. 35

Documents illustrant la pratique professionnelle (*facultatif*)

p. 36

Annexes (*Si le RC le prévoit*)

p. 37

EXEMPLES DE PRATIQUE PROFESSIONNELLE

Activité-type 1

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

Exemple n°1 ► Projet pour l'examen

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

I. Présentation du projet (vue globale)

C'est dans le cadre de ma formation à PopSchool Nancy et du projet que je vais présenter à l'examen de fin de formation que j'ai décidé de créer un site web de vidéos dédiées à l'apprentissage du piano. Ceci dans le but de mieux maîtriser les langages de programmation (HTML, CSS, JavaScript, PHP, SQL) essentiels à l'exercice du métier visé par ce titre professionnel.

Le projet se décompose en plusieurs étapes :

- La mise en place d'un cahier de charges
- La visualisation du projet avec des WireFrames
- La rédaction des spécifications fonctionnelles et techniques
- Le développement
- La phase de test
- Les évolutions potentielles

II. Gestion du projet

Dans le cadre de la gestion du projet, j'ai recueilli et suivi de nombreux conseils trouvés sur internet. Tout au long des phases préalables au développement à proprement parlé, j'ai essayé de trouver un compromis entre ce que je savais faire, ce que je pensais pouvoir apprendre à faire dans un délai raisonnable et ce que je voulais faire. J'ai donc fait évoluer les éléments qui composent mon projet conjointement, les uns influençant les autres et vice versa, pour que le résultat soit homogène et cohérent dans le but que je m'étais fixé, et ce à toutes les étapes de la conception :

- fonctionnalités nécessaire
- conception graphique
- intégration HTML/CSS
- interactivité
- gestion du contenu

III. Maquetter une application

1. Rédaction d'un cahier des charges

➤ Les contraintes d'accessibilité :

- le site web doit être conforme aux exigences et aux recommandations de WCAG (Web Content Accessibility Guidelines) et fournir un contenu : perceptible, utilisable, compréhensible et robuste.
- le site est gratuit

➤ Les contraintes ergonomiques :

- optimisation pour les mobiles, le responsive web design.
- amélioration de l'expérience utilisateur en élaborant une interface épurée, n'affichant que les éléments indispensables.

➤ Présentation des différents publics :

- le site s'adresse à toutes les personnes passionnées par le piano ou ceux qui veulent apprendre.
- l'application permet de gérer 2 profils (Administrateur, Utilisateur). Chaque profil aura des droits spécifiques. Ces droits sont définis de manière fixe et ne sont pas configurables.

➤ L'arborescence des pages du site web.

- le site doit avoir un dispositif de navigation simple et intuitif.
- liens vers les différentes catégories du site à l'affichage de la page d'accueil.
- un menu plein écran apparaît en cliquant sur un bouton de la barre de navigation (qui disparaît en scrollant vers le bas et inversement) et donne accès à toutes les pages du site ainsi qu'à différents réseaux sociaux.
- pied de page, avec les mentions légales, la charte du site, un « à propos » et liens vers différents réseaux sociaux.

2. Les wireframes

En fonction de du cahier des charges établi j'ai ensuite élaboré la structure des pages en wireframe.

J'ai décidé de suivre le principe de KISS ("Keep It Stupid and Simple") qui dit que l'interface utilisateur

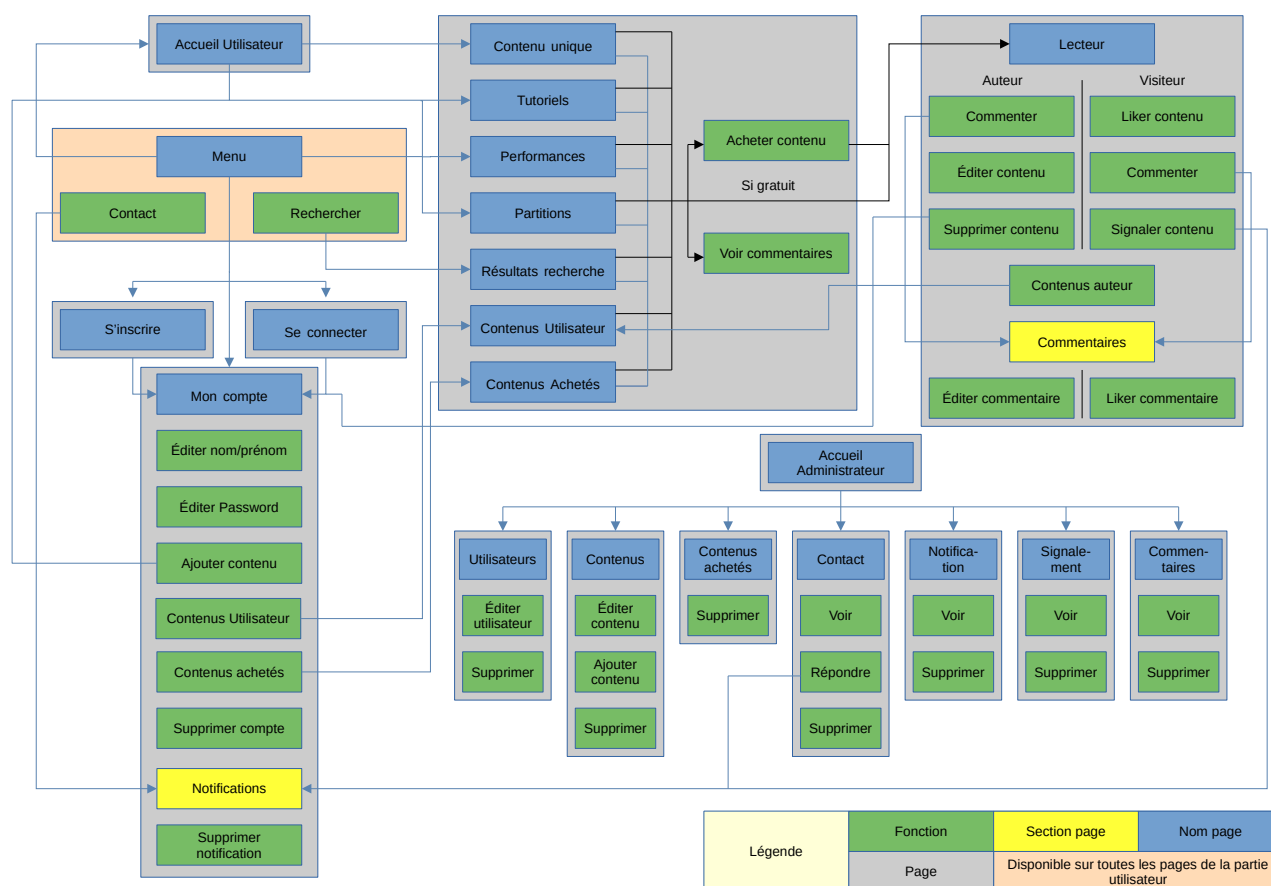
doit être la plus simple et la plus intuitive possible. Des icônes, assez grandes pour être cliquables même par des gros doigts, une mise en page qui respecte le confort utilisateur quand il navigue à une seule main, des boutons de taille suffisante, ...

J'ai alors pu mettre en place les différents éléments de navigation de façon ergonomique et proposer les wireframes des différentes pages.

Vous trouverez les wireframes de toutes les pages du site en **annexe 1 p.37**

3. Arborescence du site

En fonction du cahier des charges et des WireFrames que j'ai présenté plus haut j'ai pu établir une arborescence du squelette du site.



IV. Réaliser une interface utilisateur web statique et adaptable

1. Interface statique

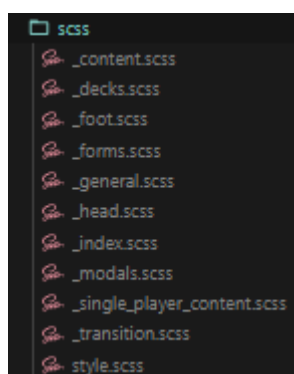
Étant daltonien et faisant le site sans aide extérieure, j'ai décidé de n'utiliser que des nuances de gris pour ne pas me retrouver avec un résultat, au niveau de la palette de couleurs utilisées, qui aurait sans été légèrement étrange. Mon principal soucis était donc de respecter les wireframes dans leur structures. Cependant, pour que le résultat ne pas trop « plat » j'ai opté pour une utilisation à peu près généralisée d'effets d'ombrage avec text-shadow, box-shadow et l'animation de nombreux éléments.

Pour mettre en place tout ceci, j'ai utilisé SASS que j'apprécie pour le fait que l'on puisse compartimenter les différentes parties du site dans des fichiers séparés et pour la notation que je trouve plus intuitive et moins laborieuse que le vanilla CSS.

Pour ce qui est des animations j'ai utilisé GSAP pour les éléments qui composent les pages et barba.js pour avoir une transition entre les pages qui sont toutes les deux des librairies javascript.

GSAP est particulièrement utile pour ses time-lines qui permettent d'éviter un travail extrêmement laborieux si l'on voulait réaliser la même chose en CSS.

Quant à barba.js il est utilisé conjointement à GSAP car il est nécessaire au chargement d'une page d'imbriquer GSAP dans des fonctions de barba.js pour que les éléments animés puissent s'afficher.



Les différents fichiers .scss du projet

```
80 .cards {
81   transform: translateY(100px) scaleY(1);
82   transform-origin: 0% 100%;
83   position: relative;
84   height: 8rem;
85
86   .titles {
87     font-size: 2rem;
88     opacity: 0;
89     transform: translateY(100px);
90     a {
91       text-decoration: none;
92       color: rgb(36, 36, 36);
93       transition: color 1s;
94       text-shadow: 0 1px 0 rgb(57, 57, 57), 0 2px 0 rgb(52, 52, 52),
95                   0 3px 0 rgb(44, 44, 44), 0 4px 0 rgb(19, 19, 19),
96                   0 5px 0 rgb(12, 12, 12), 0 6px 0 rgb(0, 0, 0),
97                   6px 7px 5px rgba(0, 0, 0, 0.4), 6px 12px 10px rgba(0, 0, 0, 0.3),
98                   6px 17px 20px rgba(0, 0, 0, 0.2), 6px 22px 30px rgba(0, 0, 0, 0.1);
99       &:hover {
100         color: rgb(67, 67, 67);
101       }
102     }
103   }
104
105   &.one {
106     display: flex;
107     flex-direction: column;
108     align-items: flex-start;
109     justify-content: space-around;
110
111     .line {
112       opacity: 0;
113       transform: translateY(100px);
114       height: 7px;
115       border-radius: 7px;
116       background: #161616;
117       box-shadow: 3px -3px 3px #090909, -3px 3px 3px #232323;
118
119       &.one {
120         width: 100%;
121       }
122
123       &.two {
124         width: 50%;
125       }
126
127       &.three {
```

Exemple de code en scss qui montre sa syntaxe particulière et quelques propriétés (aux lignes 81, 88, 89, 112, 113) utiles au fonctionnement de GSAP

```
3 gsap.config({
4   nullTargetWarn: false,
5 });
6
7 let toggle = document.querySelector(".toggle");
8 let body = document.querySelector("body");
9
10 let tl = gsap.timeline({ defaults: { ease: "power4.inOut", duration: 2 } });
11 let t1l = gsap.timeline();
12
13 Complexity is 5 Everything is cool!
14 function delay(n) {
15   n = n || 2000;
16   return new Promise((done) => {
17     setTimeout(() => {
18       done();
19     }, n);
20   });
21 }
22
23 function pageTransition() {
24   tl.set(".loading-screen", {
25     right: "-100%",
26   })
27
28   .to(".loading-screen", {
29     duration: 0.9,
30     width: "100%",
31     right: "0%",
32     ease: "Expo.easeInOut",
33   })
```

capture 1

```
94 .to(
95   ".titles, .line.one, .line.two, .line.three",
96   {
97     stagger: 0.1,
98     duration: 1.2,
99     opacity: 1,
100    y: 0,
101  },
102  "-=2"
103 )
```

capture 2

Ces trois capture d'écrans montre pour la « capture 1 » l'imbrication de GSAP et barba.js, pour la « capture 2 » comment on utilise les propriétés CSS avec GSAP (ex. lié à l'extrait de code scss précédent) et pour la « capture 3 » la fonction qui initialise barba.js et permet d'effectuer la transition entre les pages et l'animation des éléments au chargement de la page.

```
170 $(function () {
171   barba.init({
172     sync: true,
173
174     transitions: [
175       {
176         async leave(data) {
177           const done = this.async();
178
179           pageTransition();
180           await delay(1500);
181           done();
182         },
183
184         async enter(data) {
185           contentAnimation();
186         },
187
188         async once(data) {
189           contentAnimation();
190         },
191       },
192     ],
193   });
194 });
```

Capture 3

2. Interface adaptable

J'ai réalisé ce site en HTML/CSS(SCSS) et je l'ai rendu accessible sur tout type d'écran.

Les pages sont structurées selon les recommandations du W3C.

Toutes les pages s'adaptent que ce soit en mode portrait ou paysage jusqu'à un minimum de 280px.

Voici un exemple de code représentatif de la manière de procédé que j'ai employé :

```
.info_section {
  background: linear-gradient(to top, □ black 33%, transparent 100%);
  min-height: 415px;
  &.tuto,
  &.sheet,
  &.perf {
    display: flex;
    flex-direction: column;
    align-items: center;
  }
  .content_header {
    width: 100%;
    padding: 1rem;
    line-height: 2.1rem;
    padding: 0;
    margin-top: 25px;
    &.perf,
    &.tuto,
    &.sheet {
      text-align: center;
    }
  }
}
```

@media all and (max-width : 799px)

```
.info_section {
  &.tuto {
    display: flex;
    flex-direction: column;
    align-items: flex-start;
    padding-left: 3rem;
  }
  &.perf {
    display: flex;
    flex-direction: column;
    align-items: center;
  }
  &.sheet {
    display: flex;
    flex-direction: column;
    align-items: flex-end;
    padding-right: 3rem;
  }
}
.content_header {
  &.tuto {
    text-align: start;
  }
  &.perf {
    text-align: center;
  }
  &.sheet {
    text-align: end;
  }
}
```

@media all and (min-width : 800px)

V. Développer une interface utilisateur web dynamique

La plupart des pages de ce site sont dynamiques car le contenu de chacune d'entre elles varie en fonction de différents paramètres.

Pour illustrer ceci je vais simplement vous montrer un extrait du code de la page Contenu illustrant ceci :

Les contenus proposés apparaîtront sous forme de carte ayant toute la même structure (voir wireframes annexe 1 p.37 « page Contenu desktop/mobile » et « détails cartes page Contenu »). Un titre, le nom du compositeur, sa catégorie (tutoriel, performance, partition), son niveau (facile, moyen, difficile, très difficile), le nombre de likes et une mention indiquant :

- « le prix » du contenu si l'utilisateur n'a pas acheté le contenu ou que celui-ci n'est pas connecté
- « gratuit » et si c'est le cas que le contenu soit accessible sans nécessité d'être connecté
- « votre contenu » si l'utilisateur est l'auteur du contenu
- « acheté » si l'utilisateur a acheté le contenu

```
167 <main class="autoAlpha" data-barba="wrapper">
168   <div class="min-height flex_content" data-barba="container"
169     data-barba-namespace="content-section">
170     <div class="container">
171
172
173     <?php foreach ($getContents as $getContent) {
174
175         require('./assets/require/variables.php');
176
177         $getUserContentInformations = getUserContentInformations($bdd, $getContent_id_user);
178
179         require('./assets/require/variables.php');
180
181         require('./assets/require/modals_foreach.php'); ?>
182
183
184         <div class="box">
185             <div class="card">
186                 <figure class="card_thumb">
187                     </img>
```

Ligne 173 : récupération de tous les contenus demandés par l'utilisateur

Ligne 177 : récupération des informations sur chaque contenus récupérés ligne 173

```
190 <figcaption class="card_caption">
191
192 <h2 class="card_title"><?= $getContent_title ?></h2>
193 <h2 class="card_composer"><?= $getContent_composer ?></h2>
194 <div class="card_likes"><i class="far fa-thumbs-up"> <?= $getContent_likes ?></i></div>
195 <span class="card_category"><?= $getContent_category ?></span>
196 <span class="card_level"><?= $getContent_level ?></span>
197 <p class="card_snippet"></p>
198
199
200 <?php if (isset($getUserInformations_id) && !empty($getUserInformations_id)) {
201
202
203     $getIdUserFromPurchasedContent = getIdUserFromPurchasedContent($bdd,
204     $getContent_id);
205
206     $user_session_purchased_content = in_array($getUserInformations_id, array_column
207     ($getIdUserFromPurchasedContent, 'id_users'));
208
209     if ($getContent_price == 0 || $getContent_id_user == $getUserInformations_id) {
210
211
212         if ($getContent_price == 0 && $getContent_id_user != $getUserInformations_id) { ?>
213
214
215             <div class="card_price">Free</div>
216
217
218             <?php } else if ($getContent_id_user == $getUserInformations_id) { ?>
219
220
221                 <div class="card_price">Your content</div>
222
223
224             <?php } ?>
225
226             <div class="content_button_flex">
227                 <a href="single_player_content.php?id=<?= $getContent_id ?>"
228                 class="card_button link_page">Watch</a>
229                 <a href="content.php?id=<?= $getContent_id ?>&name=visitor&
                category=user_content" class="card_button link_page">By <?=
                $getUserContentInformations_name ?> <?= $getUserContentInformations_lastname ?
                ></a>
                </div>
```

Ligne 192 à 196 : affichage des informations propres à chaque contenus demandés

Ligne 200 à 205 : vérification qu'un utilisateur est connecté. Si c'est la cas, on vérifie quels sont les contenus qu'il a acheté.

Ligne 209 : si le contenu est gratuit ou que l'utilisateur connecté est l'auteur du contenu

Ligne 212 : si le contenu est gratuit est que l'utilisateur connecté n'est pas l'auteur du contenu on affiche « free » et il peut le regarder.

Ligne 218 : si le l'utilisateur connecté est l'auteur du contenu on affiche « your content » et il peut le regarder

```
231
232     <?php } eLse { ?>
233
234
235     <?php if ($user_session_purchased_content == false) { ?>
236
237
238         <div class="card_price"><?= $getContent_price ?> Credits</div>
239
240         <div class="content_button_flex">
241             <button class="card_button pointer" id="buy_button<?= $getContent_id ?>" onfocus="javascript: modalForeach ('buy','<?= $getContent_id ?>')">Watch</button>
242             <a href="content.php?id=<?= $getContent_id ?>&name=visitor&category=user_content" class="card_button link_page">By <?= $getUserContentInformations_name ?> <?= $getUserContentInformations_lastname ?></a>
243         </div>
244
245     <?php } eLse if ($user_session_purchased_content == true) { ?>
246
247
248         <div class="card_price">Purchased</div>
249
250         <div class="content_button_flex">
251             <a href="single_player_content.php?id=<?= $getContent_id ?>" class="card_button link_page">Watch</a>
252             <a href="content.php?id=<?= $getContent_id ?>&name=visitor&category=user_content" class="card_button link_page">By <?= $getUserContentInformations_name ?> <?= $getUserContentInformations_lastname ?></a>
253         </div>
254
255     <?php } ?>
256
257
258     <?php } ?>
259
260
261     <?php } eLse { ?>
262
```

Ligne 232 : si le contenu n'est pas gratuit et qu'un utilisateur est connecté

Ligne 235 : si l'utilisateur connecté n'a pas acheté le contenu on affiche « son prix » et lui on propose de l'acheter

Ligne 245 : si l'utilisateur connecté a acheté le contenu on affiche « purchased » et il peut le regarder

Ligne 261 : si l'utilisateur n'est pas connecté


```
264 <?php if ($getContent_price > 0) { ?>
265
266
267     <div class="card_price"><?= $getContent_price ?> Credits</div>
268
269     <div class="content_button_flex">
270         <button class="card_button pointer" id="buy_button"<?=
271             $getContent_id ?>" onfocus="javascript: modalForeach('buy',
272                 '<?= $getContent_id ?>')">Watch</button>
273         <a href="content.php?id=<?= $getContent_id ?>&name=visitor&
274             category=user_content" class="card_button link_page">By <?
275             = $getUserContentInformations_name ?> <?=
276             $getUserContentInformations_lastname ?></a>
277     </div>
278
279 <?php } eLse { ?>
280
281     <div class="card_price">Free</div>
282
283     <div class="content_button_flex">
284         <a href="single_player_content.php?id=<?= $getContent_id ?
285             >" class="card_button link_page">Watch</a>
286         <a href="content.php?id=<?= $getContent_id ?>&name=visitor&
287             category=user_content" class="card_button link_page">By <?
288             = $getUserContentInformations_name ?> <?=
289             $getUserContentInformations_lastname ?></a>
290     </div>
291
292 <?php } ?>
293
294 <?php ?>
295
296     </figcaption>
297 </figure>
298 </div>
299 </div>
300 </main>
301
```

Ligne 264 : si le contenu n'est pas gratuit on affiche le prix du contenu et on propose à l'utilisateur non connecté de se connecter ou de créer un compte

Ligne 274 : si le contenu est gratuit on affiche « free » et l'utilisateur non connecté peut le regarder

VI. Réaliser une interface utilisateur avec une solution de gestion de contenu

Comme vous avez pu le constater dans les points précédant, il apparaît que ce projet répond, je pense, à ce que l'on peut assimiler à une interface utilisateur avec solution de gestion de contenu car :

- un utilisateur peut créer un compte, se connecter, supprimer son compte
- il est possible de modifier son nom, prénom, mot de passe
- on peut poster des contenus, en modifier tous les éléments, les supprimer, les liker ou les signaler quand on en est pas l'auteur
- on peut poster des commentaires, les modifier, les liker quand on en est pas l'auteur
- on peut contacter un administrateur
- on peut recevoir des notifications pour diverses raisons

VII. La phase de test de la partie front-end

Pour obtenir un bon résultat en ce qui concerne le web responsive design j'ai travaillé avec :

- des navigateurs différents (chrome, edge, firefox, opéra)
- des tailles d'écrans différentes
- les outils disponibles dans les consoles des navigateurs pour simuler des événements tactiles etc..
- le validateur du W3C (<https://validator.w3.org/>) pour l'HTML
- le validateur (<https://jigsaw.w3.org/css-validator/>) pour le CSS

VIII. Sécurité

Les mesures prises pour la sécurité seront traitées dans l'exemple n°2 qui traite de la partie back-end du projet car bien que des mesures de sécurité existent du côté front-end elles sont complémentaires à ce qui a été mis en place du côté back-end qui est la pierre angulaire en ce qui concerne la sécurité d'un site web. Donc pour éviter les répétitions, je vous invite à consulter cette section.

Voici néanmoins quelques éléments essentiels que nous pouvons citer dans cette partie :

- Toutes les parties du site où des actions peuvent influencer sur le contenu du site sont protégées par l'authentification.
- Tous les entrées utilisateurs sont vérifiées côté front et côté back et, quand elles sont affichées, les variables sont échappées avec `htmlspecialchars`.

DOSSIER PROFESSIONNEL (DP)

2. Précisez les moyens utilisés :

Pour la partie front-end de ce projet j'ai utilisé les ressources suivantes :

- Visual Studio Code
- HTML, CSS, Bootstrap pour la partie administrateur du site, JavaScript, les librairies Javascript GSAP et barba.js
- <https://developer.mozilla.org/>

3. Avec qui avez-vous travaillé ?

J'ai travaillé seul sur ce projet

4. Contexte

Nom de l'entreprise, organisme ou association ► PopSchool Nancy

Chantier, atelier, service ►

Période d'exercice ► Du : 10/01/2022 au : 05/08/2022

5. Informations complémentaires (facultatif)

Activité-type

2

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

Exemple n° 2 ► Projet pour l'examen

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

I. Établir les besoins

Dès la phase de conception de ce projet que j'ai décrit dans les premiers chapitres de l'activité-type n°1, il était évident qu'il serait nécessaire de mettre en place une partie back-end pour arriver au résultat que je visais.

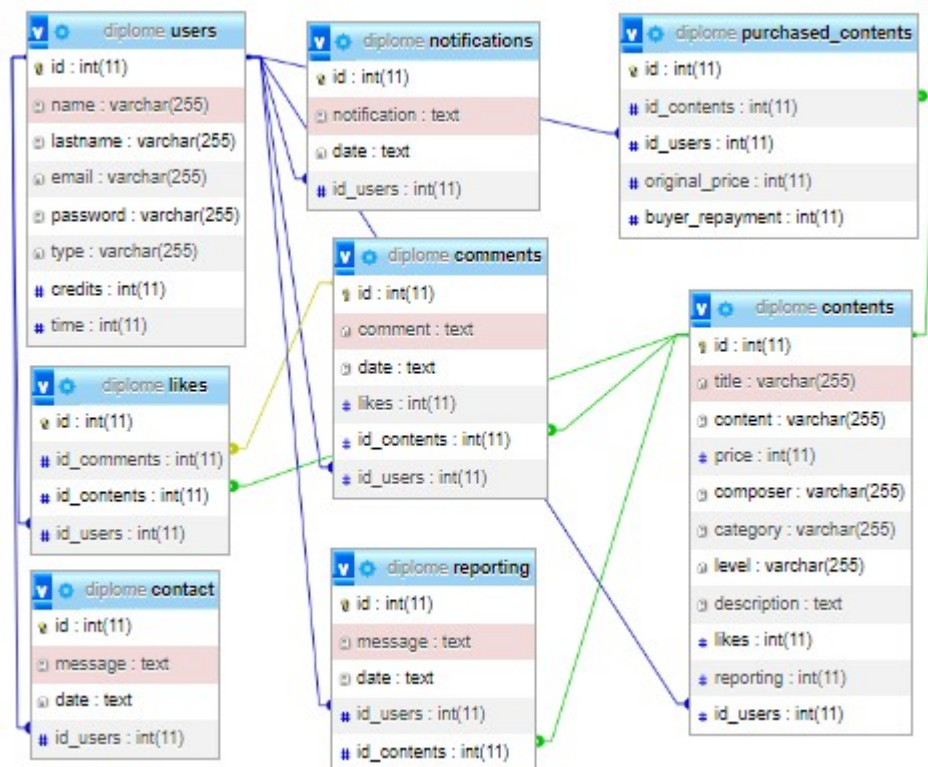
Pour cela j'avais besoin :

- de créer une base de données
- de définir par rapport aux éléments front-end mis en place et aux fonctionnalités qui en découlent naturellement, le nombre de tables et d'entrées qui les composent
- de structurer le projet de manière à compartimenter au maximum les différents éléments essentiellement dans un soucis de clarté et de maintenabilité du code
- d'intégrer dans chaque fichiers (front ou back) des éléments destinés à la sécurité du site

II. Conception et création d'une base de données

J'ai créé la base de donnée « diplôme » de mon projet à partir de l'interface graphique en ligne phpMyAdmin. Je m'y suis connecté comme l'utilisateur root (administrateur), puis j'ai créé la base avec l'encodage utf8_general_ci, dont voici la représentation :

DOSSIER PROFESSIONNEL (DP)



Script SQL en annexe 2 p.46

III. Développer les composants d'accès aux données

Pour accéder aux données j'utilise plusieurs fichiers ayant chacun un rôle différents. Je vais vous détailler le cheminement des données effectué par l'intermédiaire de ces fichiers, de la base de donnée jusqu'à l'affichage sur la page. Pour cela, imaginons un utilisateur arrivant sur son compte et voulant accéder au contenus qu'il a acheté :

- clique sur le bouton permettant d'accéder aux contenus achetés

```
<button class="button gray"><a class="link_page" href="content.php?category=user_purchased_content">Purchased Content</a></button>
```

- redirection vers la categorie « user_purchased_content » de la page **content.php**

```
<?php } else if ($page == 'user_purchased_content') {  
  
    $getContent = getUserPurchasedContent($bdd, $session_users_id);  
  
    if (empty($getContent)) {  
        $bdd = null;  
        header('location: /Diplome/my_account.php?error=00212');  
        die();  
    } ?>
```

- appel de la fonction **getUserPurchasedContent()**, qui va récupérer tous les contenus achetés par un utilisateur, située dans le fichier **functions.php** ou redirection vers **my_account.php** avec un message d'erreur informant l'utilisateur qu'il n'a pas acheté de contenu.

Remarque : toutes les redirections ont un code d'erreur ou de succès unique, cela permet d'une part de renseigner l'utilisateur sur le fonctionnement de certaines choses et d'autre part cela facilite les investigations dans le cadre d'un débogage.

```
function getUserPurchasedContent(PDO $bdd, $session_users_id)  
{  
    $req = $bdd->prepare('SELECT purchased_contents.id_contents, contents.title , contents.  
composer, contents.category, contents.content, contents.price, contents.id, contents.  
description, contents.id_users, contents.likes, contents.level  
FROM purchased_contents  
INNER JOIN contents  
ON purchased_contents.id_contents = contents.id  
WHERE purchased_contents.id_users = :id_users ');  
    $req->bindParam(':id_users', $session_users_id, PDO::PARAM_INT);  
    $req->execute();  
    $contents = $req->fetchAll();  
    return $contents;  
}
```

- connexion à la base de donnée avec PDO

```
try {  
    $bdd = new PDO('mysql:host=localhost;dbname=diplome;charset=UTF8', 'root', '', [  
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,  
        PDO::ATTR_PERSISTENT => true,  
        PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,  
    ]);  
} catch (PDOException $e) {  
    error_log('PDOException - ' . $e->getMessage(), 0);  
    http_response_code(500);  
    die('Error establishing connection with database');  
}
```

- stockage des données dans la variable **\$getContents** vue plus haut

- appel de cette variable dans un **foreach** pour parcourir l'**array** généré par la fonction **getUserPurchasedContent()**

```
<?php foreach ($getContents as $getContent) {  
    require('./assets/require/variables.php');
```

- appel du fichier **variables.php** dont la fonction est de regrouper toutes les variables destinées à être affichées et d'y faire les traitements nécessaires à la sécurité du site et d'alléger le code des pages

```
if (isset($getContent) && !empty($getContent)) {  
  
    $getContent_id = htmlspecialchars($getContent['id']);  
    $getContent_title = htmlspecialchars($getContent['title']);  
    $getContent_composer = htmlspecialchars($getContent['composer']);  
    $getContent_category = htmlspecialchars($getContent['category']);  
    $getContent_level = htmlspecialchars($getContent['level']);  
    $getContent_video = htmlspecialchars($getContent['content']);  
    $getContent_image = explode('.', $getContent_video);  
    $getContent_price = htmlspecialchars($getContent['price']);  
    $getContent_description = str_replace('<br />', '', nl2br(htmlspecialchars($getContent['description'])));  
    $getContent_likes = htmlspecialchars($getContent['likes']);  
    $getContent_id_user = htmlspecialchars($getContent['id_users']);  
}
```

- affichage des éléments demandés

```
<h2 class="card_title"><?= $getContent_title ?></h2>  
<h2 class="card_composer"><?= $getContent_composer ?></h2>  
<div class="card_likes"><i class="far fa-thumbs-up"> <?= $getContent_likes ?></i></div>  
<span class="card_category"><?= $getContent_category ?></span>  
    <span class="card_level"><?= $getContent_level ?></span>
```

IV. Développer la partie back-end d'une application web ou web mobile

Je vais maintenant dans les détails la manière dont j'ai structuré une page. Toutes les pages suivant la même logique je ne vous en présenterai qu'une. Je ne détaillerai cependant pas tous les éléments (seulement ceux qui me paraissent intéressants) pour vous épargner les répétitions.

```
1  <?php
2  session_start();
3  require('./assets/require/check_data.php');
4
5  if (
6      isset($get_id)
7      && (isset($session_users_id) xor !isset($check_session_users_id))
8      && (isset($get_error) xor !isset($check_get_error))
9      && (isset($get_success) xor !isset($check_get_success))
10 ) {
11
12
13     $page = 'single_player';
14     require('./assets/require/co_bdd.php');
15     require('./assets/require/functions.php');
16
17     $getContentAndUserInformations = getContentAndUserInformations($bdd, $get_id);
```

Ligne 2 : il peut être nécessaire d'être connecté pour aller sur cette page mais ce n'est pas obligatoire

Ligne 3 : appel du fichier check_data.php qui sert à vérifier si les données nécessaires pour aller sur une page ont le format attendu :

Par exemple le `isset($get_id)` de la ligne 6 :

```
101  if (isset($_GET['id']) && !empty($_GET['id'])) {
102
103      $check_get_id = is_numeric($_GET['id'])
104          && preg_match("/^[0-9]+$/", $_GET['id']);
105
106      if ($check_get_id === true) {
107
108          $get_id = $_GET['id'];
109      }
110  }
```

Les détails sur cette manière de faire seront développés dans le chapitre sur la sécurité p.

Ligne 7 : Ici avec la même manière de procéder que pour `$get_id` vérification qu'un utilisateur est bien connecté et que la variable a bien le format attendu ou qu'elle n'existe pas du tout. *Remarque : pour la cohérence du code toutes ces variables sont nommées avec la même logique. Ici `$session_user_id` correspond à `$_SESSION['user']['id']`. Cela permet également d'améliorer la lisibilité du code.*

Ligne 10 : Si les données n'ont pas le format attendu une redirection vers la page d'accueil est effectuée.

```
237 <?php } else {  
238  
239     $bdd = null;  
240     header('location: index.php?error=00615');  
241     die();  
242 } ?>
```

Une éventuelle connexion à la base de donnée est close, redirection avec une erreur « Une erreur est survenue » (toutes les erreurs ont un code unique qui affiche une modale avec le message correspondant) et le script courant est terminé.

Ligne 13 : La variable \$page sert à modifier les liens du menu en fonction de la page sur laquelle on se trouve.

Ligne 14 : Appel du fichier de connexion à la base de données.

Ligne 15 : Appel du fichier **function.php** contenant toutes les fonctions utilisées sur les pages :

Par exemple **getContentAndUserInformations** de la ligne 17 qui sert ici à récupérer les informations liées au contenu.

```
145 function getContentAndUserInformations(PDO $bdd, $get_id)  
146 {  
147     $req = $bdd->prepare('SELECT users.name, users.lastname, contents.id, contents.title,  
148         contents.composer, contents.category, contents.level, contents.content, contents.price,  
149         contents.description, contents.likes, contents.reporting, contents.id_users  
150     FROM users  
151     INNER JOIN contents  
152     ON users.id = contents.id_users WHERE contents.id = :contents_id ');  
153     $req->bindParam(':contents_id', $get_id, PDO::PARAM_INT);  
154     $req->execute();  
155     $content = $req->fetch();  
156     return $content;  
157 }
```



```
20 if ($getContentAndUserInformations) {
21
22     require('./assets/require/variables.php');
23
24     $getComments = getComments($bdd, $get_id);
25
26
27     if (isset($session_users_id) && !empty($session_users_id)) {
28
29         $getUserInformations = getUserInformations($bdd, $session_users_id);
30
31         require('./assets/require/variables.php');
32
33         $getIdUserFromPurchasedContent = getIdUserFromPurchasedContent($bdd,
34             $getContentAndUserInformations_id);
35
36         $user_session_purchased_content = in_array($getUserInformations_id, array_column(
37             $getIdUserFromPurchasedContent, 'id_users'));
38     }
39
40     if (($getContentAndUserInformations_price > 0 && isset($session_users_id)) ||
41         $getContentAndUserInformations_price == 0) {
42
43         if ((isset($user_session_purchased_content) && $user_session_purchased_content ==
44             true) || $getContentAndUserInformations_price == 0 ||
45             $getContentAndUserInformations_id_user == $getUserInformations_id) {
46
47             require('./assets/require/head.php'); ?>
```

Ligne 20 : Vérification de l'existence du contenu sinon redirection vers la page d'accueil avec une erreur « Ce contenu n'existe pas ».

Ligne 22 : Appel du fichier **variables.php** qui regroupe toutes les variables appelées

Par exemple **getContentAndUserInformations** de la ligne 20 :

```
94 if (isset($getContentAndUserInformations) && !empty($getContentAndUserInformations)) {
95
96     $getContentAndUserInformations_id = htmlspecialchars($getContentAndUserInformations['id']);
97     $getContentAndUserInformations_title = htmlspecialchars($getContentAndUserInformations['title']);
98     $getContentAndUserInformations_composer = htmlspecialchars($getContentAndUserInformations['composer']);
99     $getContentAndUserInformations_category = htmlspecialchars($getContentAndUserInformations['category']);
100    $getContentAndUserInformations_level = htmlspecialchars($getContentAndUserInformations['level']);
101    $getContentAndUserInformations_video = htmlspecialchars($getContentAndUserInformations['content']);
102    $getContentAndUserInformations_price = htmlspecialchars($getContentAndUserInformations['price']);
103    $getContentAndUserInformations_description = str_replace('<br />', ' ', nl2br(htmlspecialchars(
104        $getContentAndUserInformations['description'])));
105    $getContentAndUserInformations_likes = htmlspecialchars($getContentAndUserInformations['likes']);
106    $getContentAndUserInformations_id_user = htmlspecialchars($getContentAndUserInformations['id_users']);
107    $getContentAndUserInformations_author_name = htmlspecialchars($getContentAndUserInformations['name']);
108    $getContentAndUserInformations_author_lastname = htmlspecialchars($getContentAndUserInformations
109        ['lastname']);
110 }
```

Dans ce fichier les différents traitements nécessaires pour afficher des informations dans une page sont appliqués. *Remarque: les variables sont toutes nommées avec la même logique. Ici on sait par exemple que `$getContentAndUserInformations_id` est issu de la variable `$getContentAndUserInformations` qui est elle-même générée par la fonction `getContentAndUserInformations()`. Cela permet donc une nouvelle fois d'alléger le code et de savoir dès que l'on rencontre une variable dans une page de savoir d'où elle vient.*

Ligne 33: Récupération de l'`id` de tous les utilisateurs ayant acheté ce contenu.

Ligne 35 : L'`id` de l'utilisateur connecté fait-elle partie des `id` récupéré à la ligne 33.

Ligne 39 : Si le contenu n'est pas gratuit il faut être connecté sinon redirection vers page d'accueil avec une erreur « Ce contenu n'est pas gratuit . Connectez-vous ou inscrivez-vous. Vous obtiendrez 500 Crédits ».

Ligne 41 : Si l'utilisateur connecté a acheté le contenu ou si il est gratuit ou si l'utilisateur connecté est l'auteur du contenu, visionnage autorisé sinon redirection vers page d'accueil avec une erreur « Vous n'avez pas acheté ce contenu ».

Ligne 43 : Début du chargement de la page avec l'appel de `head.php` qui contient hormis le `!doctype` etc. l'appel de deux fichiers nécessaire à la sécurité du site `page_deco_auto.php` qui déconnecte l'utilisateur automatiquement au bout de 15 min en cas d'inactivité et `session_regenerate.php` qui régénère l'identifiant de session toutes les 5 min. Les détails concernant ces fichiers seront développés dans le chapitre sur la sécurité.

```
46 <main class="autoAlpha" data-barba="wrapper">
47   <div class="min-height" data-barba="container"
48     data-barba-namespace="single_player_content-section">
49     <div class="movie-card">
50       <div class="single_player_container">
51         <div class="hero">
52           <video src="./assets/videos/<?= $getContentAndUserInformations_video ?>"
53             controls controlsList="nodownload"></video>
54           <div class="details">
55             <h1 class="title1"><?= $getContentAndUserInformations_title ?></h1>
56             <h2 class="title2"><?= $getContentAndUserInformations_composer ?></h2>
57             <h2 class="title3"><?= $getContentAndUserInformations_category ?> <?=
58               $getContentAndUserInformations_level ?></h2>
59           </div>
60         </div>
61       </div>
62     <div class="player_bottom_bar">
63       <span class="likes"><i class="far fa-thumbs-up"> <?=
64         $getContentAndUserInformations_likes ?></i></span>
65
```

Ligne 46 à 64 : Affichage de la vidéo et de différentes informations. Rien de particulier si ce n'est la présence des data-barba « wrapper » et « container » nécessaires aux transitions entre les pages.

```
76 <?php if (isset($getUserInformations_id) && !empty($getUserInformations_id))
77 {
78
79     if ($getUserInformations_id != $getContentAndUserInformations_id_user) { ?>
80
81         <button class="dropbtn gray" id="like_content_button"
82         onfocus="javascript:modal('like_content')">Like</button>
83
84         <div class="dropdown">
85             <button class="drop">Comment/Report</button>
86             <div class="dropdown-content">
87                 <button class="button_drop gray" id="comment_button"
88                 onfocus="javascript:modal('comment')">Comment</button>
89                 <button class="button_drop gray" id="report_button"
90                 onfocus="javascript:modal('report')">Report this Content</button>
91             </div>
92         </div>
93
94     <?php } else { ?>
95
96         <button class="dropbtn gray" id="comment_button"
97         onfocus="javascript:modal('comment')">Comment</button>
98
99         <div class="dropdown">
100             <button class="drop">Edit/Delete</button>
101             <div class="dropdown-content">
102                 <button class="button_drop gray" id="edit_button"
103                 onfocus="javascript:modal('edit')">Edit Content</button>
104                 <button class="button_drop gray" id="delete_content_button"
105                 onfocus="javascript:modal('delete_content')">Delete Content</button>
106             </div>
107         </div>
108
109     <?php } ?>
110
111 <?php } ?>
```

Ligne 76 : Si l'utilisateur est connecté

Ligne 79 : Si l'utilisateur connecté n'est pas l'auteur du contenu affichage des boutons « Like », « Comment » et « Report this content » ouvrant chacun des modales.

Ligne 93 : Si l'utilisateur connecté est l'auteur du contenu affichage des boutons « Comment », « Edit content » et « Delete content » ouvrant chacun des modales.

```
112
113
114
115
116
117
118
119
120
121
122
123
    </div>
    <div class="description">
        <div class="column1 gray">
            <a class="link_page" href="content.php?id=<?=
            $getContentAndUserInformations_id ?>&name=visitor&category=user_content"><?
            = $getContentAndUserInformations_author_name . " " .
            $getContentAndUserInformations_author_lastname ?></a>
        </div>
        <div class="column2">
            <p><?= $getContentAndUserInformations_description ?></p>
        </div>
    </div>
```

Affichage du nom de l'auteur dans un lien conduisant vers la page des contenus de cet utilisateur et affichage de la description du contenu.

```
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
    <?php foreach ($getComments as $getComment) {
        require('./assets/require/variables.php');
        $getNumbersOfComments = getNumbersOfComments($bdd, $getComment_user_id);
        require('./assets/require/variables.php');
        require('./assets/require/modals_foreach.php'); ?>
        <div class='deck'>
            <div class='single_player_card'>
                <div class='cardHeader'>
                    <span class='cardHeader_account'><?= $getComment_user_name . " " .
                    $getComment_user_lastname ?></span>
                    <span class='cardHeader_date'><?= $getComment_date ?></span>
                </div>
                <div class='cardBody'>
                    <p class='cardText'><?= $getComment_text ?></p>
```

Ligne 126 : Affichage de tous les commentaires liés au contenu, récupérés par la fonction `getComments()` ligne 24

```
157 function getComments(PDO $bdd, $get_id)
158 {
159     $req = $bdd->prepare('SELECT users.name, users.lastname, comments.comment, comments.id,
160     comments.id_users, comments.date, comments.likes
161     FROM comments
162     INNER JOIN contents
163     ON comments.id_contents = contents.id
164     INNER JOIN users
165     ON comments.id_users = users.id
166     WHERE comments.id_contents = :contents_id ORDER BY comments.date ASC');
167     $req->bindParam(':contents_id', $get_id, PDO::PARAM_INT);
168     $req->execute();
169     $comments = $req->fetchAll();
170     return $comments;
171 }
```

Ligne 130 : Récupération du nombres de commentaires postés par l’auteur du commentaire.

Ligne 134 : Appel d’un fichier contenant les modales destinés à être appelées dans un foreach.

```
149 <section class='cardStats'>
150
151     <span class='cardStats_stat cardStats_stat-comments'><?=
152     $getNumbersOfcomments_from_user ?> <i class='far fa-comment
153     fa-fw'></i></span>
154
155     <span class='cardStats_stat cardStats_stat-likes'><?=
156     $getComment_likes ?> <i class='far fa-thumbs-up'></i></span>
157
158 </section>
159 </div>
160
161 <?php if (isset($session_users_id) && ($session_users_id !=
162 $getComment_user_id)) { ?>
163
164     <button class="dropbtn gray" id="like_comment_button"><?=
165     $getComment_id ?>" onfocus="javascript: modalForeach('like_comment',
166     '<?= $getComment_id ?>')">Like</button>
167
168 <?php } eLse if (isset($session_users_id) && ($session_users_id ==
169 $getComment_user_id)) { ?>
170
171     <button class="dropbtn gray" id="edit_comment_button"><?=
172     $getComment_id ?>" onfocus="javascript: modalForeach('edit_comment',
173     '<?= $getComment_id ?>')">Edit</button>
174
175 <?php } ?>
```

```
172
173
174         </div>
175     </div>
176
177
178     <?php } ?>
179
180
181     </div>
182 </div>
183 </div>
184 </main>
185
186 <?php require('./assets/require/footer.php'); ?>
```

Affiche la partie statistique de la carte de commentaire (le nombre de likes et le nombres de commentaires postés par l’auteur) ainsi qu’un bouton « Edit » pour éditer le commentaire ou un bouton « like » pour le liker suivant que l’on est l’auteur ou non du commentaire. Ces deux boutons ouvrent chacun une modale.

Ligne 186 : Appel du footer.

La page se termine ensuite par les différentes redirections qui ont été mentionnées précédemment .

V. Mettre en œuvre des composants dans une application de gestion de contenu

Comme vous avez pu le constater dans les points précédant, il apparaît que ce projet répond, je pense, à ce que l’on peut assimiler à une interface utilisateur avec solution de gestion de contenu car :

- un utilisateur peut créer un compte, se connecter, supprimer son compte
- il est possible de modifier son nom, prénom, mot de passe
- on peut poster des contenus, en modifier tous les élément, les supprimer, les liker ou les signaler quand on en est pas l’auteur
- on peut poster des commentaires, les modifier, les liker quand on en est pas l’auteur
- on peut contacter un administrateur
- on peut recevoir des notifications pour diverses raisons

VI. La phase de test de la partie back-end

Pour chaque nouvelle fonctionnalité implémenté lors de la création de la partie back-end, j’ai réalisé des tests en essayant de déterminer tous les comportements (conventionnels ou malveillants) qu’un utilisateur pouvait avoir. J’ai ensuite fait évoluer, en fonction des résultats des tests, le code pour remédier au maximum de problèmes.

VII. Sécurité

Tout au long de ce projet, j'ai essayé de mettre en oeuvre de bonnes pratiques de sécurité, et d'effectuer une veille sur les éventuelles vulnérabilités de sécurité de mon projet. Voici une liste des principaux points que j'ai mis en place grâce à cette veille :

- en premier lieu, on peut citer la différenciation, au niveau de la base de donnée, des simples utilisateurs, des administrateurs

id	name	lastname	email	password	type	credits	time
2	User	One	userone@gmail.com	\$2y\$10\$BMzTaCZErf0OPf0y6OzsJau80fUUypsdUZLdglx...	admin	606	1665599972
3	User	Two	usertwo@gmail.com	\$2y\$10\$Mw1OOEEv4i9cwzXQHVRs3emJrS2JDF7IXU6jLcNI6KC...	user	400	1665585804

- les mots de passe sont hashés en base de données

```
$password = password_hash($post_password, PASSWORD_BCRYPT);
```

- déconnexion automatique au bout de 15 min

```
3  if (isset($session_users_id)) {
4
5      $req = $bdd->prepare('SELECT time FROM users WHERE id= :session_users_id');
6      $req->bindParam(':session_users_id', $session_users_id);
7      $req->execute();
8      $time = $req->fetch();
9
10     $time = intval(implode($time));
11
12     if ($time + 900 - time() <= 0) {
13
14         unset($_SESSION['users']);
15         session_destroy();
16         $bdd = null;
17         header('location: index.php?success=007246');
18         die();
19     }
20 }
```

- l'identifiant de session est renouvelé toutes les 5 minutes (une fréquence de renouvellement plus élevée peut entraîner des problèmes sur mobile liés à une plus grande instabilité des réseaux) pour prévenir la fixation de session et dans une certaine mesure les attaques CSRF (dans une certaine mesure car je ne régénère l'identifiant que toutes les 5 minutes pour la raison susmentionnée).

- l'identifiant de session est renouvelé toutes les 5 minutes (une fréquence de renouvellement plus élevée peut entraîner des problèmes sur mobile liés à une plus grande instabilité des réseaux) pour prévenir la fixation de session et dans une certaine mesure les attaques CSRF (dans une certaine mesure car je ne régénère l'identifiant que toutes les 5 minutes pour la raison susmentionnée).

```
1 <?php
2 if (isset($session_users_id)) {
3
4     $old_session = session_id();
5
6     $req = $bdd->prepare('SELECT time FROM users WHERE id= :session_users_id');
7     $req->bindParam(':session_users_id', $session_users_id);
8     $req->execute();
9     $time = $req->fetch();
10
11     $time = intval(implode($time));
12
13     if ($time + 300 - time() <= 0) {
14
15         $time = time();
16
17         $req = $bdd->prepare('UPDATE users SET time = :time WHERE id= :session_users_id');
18         $req->bindParam(':time', $time);
19         $req->bindParam(':session_users_id', $session_users_id);
20         $req->execute();
21
22         session_regenerate_id();
23         $new_session = session_id();
24
25         if (isset($new_session)) {
26
27             session_write_close();
28             session_id($new_session);
29         } else {
30
31             session_write_close();
32             session_id($old_session);
33         }
34     }
35 }
```

- pour la prévention des failles XSS toutes les variables appelées sont entourées par **htmlspecialchars** comme vous pouvez le voir avec cet exemple p.25 et sont contenu dans un seul fichier **variables.php** pour alléger la lecture du code et faciliter la modification. *Remarque : comme vous pourrez le constater dans le point suivant, les données en entrée ne sont pas nettoyées en échappant ou en supprimant des caractères car des attaquants savent contourner cette protection. La meilleure façon de se protéger des failles XSS est donc de convertir les caractères spéciaux en entité html en utilisant htmlspecialchars.*

DOSSIER PROFESSIONNEL (DP)

- les entrées utilisateur sont filtrées de la manière la plus restrictive possible du côté client et du côté serveur comme par exemple p.23. Ceci vient donc en complément du point précédant pour la prévention des failles XSS et du point suivant pour la prévention des injections SQL (bien que la validation des entrées utilisateur ne soit pas une solution directe à l'injection SQL, elle aide à éviter que des données utilisateur malveillantes ne soient interprétées par la base de données). Cette façon de procéder permet de s'assurer que les données ont bien le format attendu, de ne pas dénaturer des données inoffensives par erreur en supprimant des caractères ou de ne pas augmenter la place prise par des données en échappant des caractères préalablement à leur entrée dans la base.

- pour l'upload d'un fichier une vérification d'une éventuelle double extension est réalisée pour éviter qu'un attaquant puisse transmettre un fichier avec un nom du genre **filename.php.mp4** qui pourrait s'exécuter sur le serveur comme un fichier php et engendrer toute une somme de désagréments.

2. Précisez les moyens utilisés :

Pour partie back-end du projet j'ai utilisé les ressources suivantes :

- VsCode
- Un serveur WEB Apache : Wamp chez moi
- PHP
- MySQL
- PhpMyAdmin
- Php.net
- stackoverflow.com

3. Avec qui avez-vous travaillé ?

J'ai travaillé seul sur ce projet

4. Contexte

Nom de l'entreprise, organisme ou association ► PopSchool

Chantier, atelier, service ►

Période d'exercice ► Du : 10/01/2022 au : 05/08/2022

DOSSIER PROFESSIONNEL (DP)

5. Informations complémentaires (facultatif)

Titres, diplômes, CQP, attestations de formation

(facultatif)

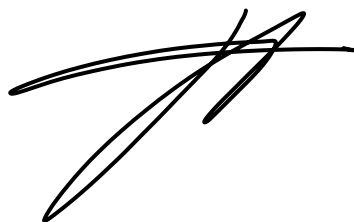
Intitulé	Autorité ou organisme	Date
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.

Déclaration sur l'honneur

Je soussigné(e) [prénom et nom] Christophe MECKLER..... ,
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 Dombasle sur Meurthe..... le 20/10/2022.....
pour faire valoir ce que de droit.

Signature :



DOSSIER PROFESSIONNEL (DP)

Documents illustrant la pratique professionnelle

(facultatif)

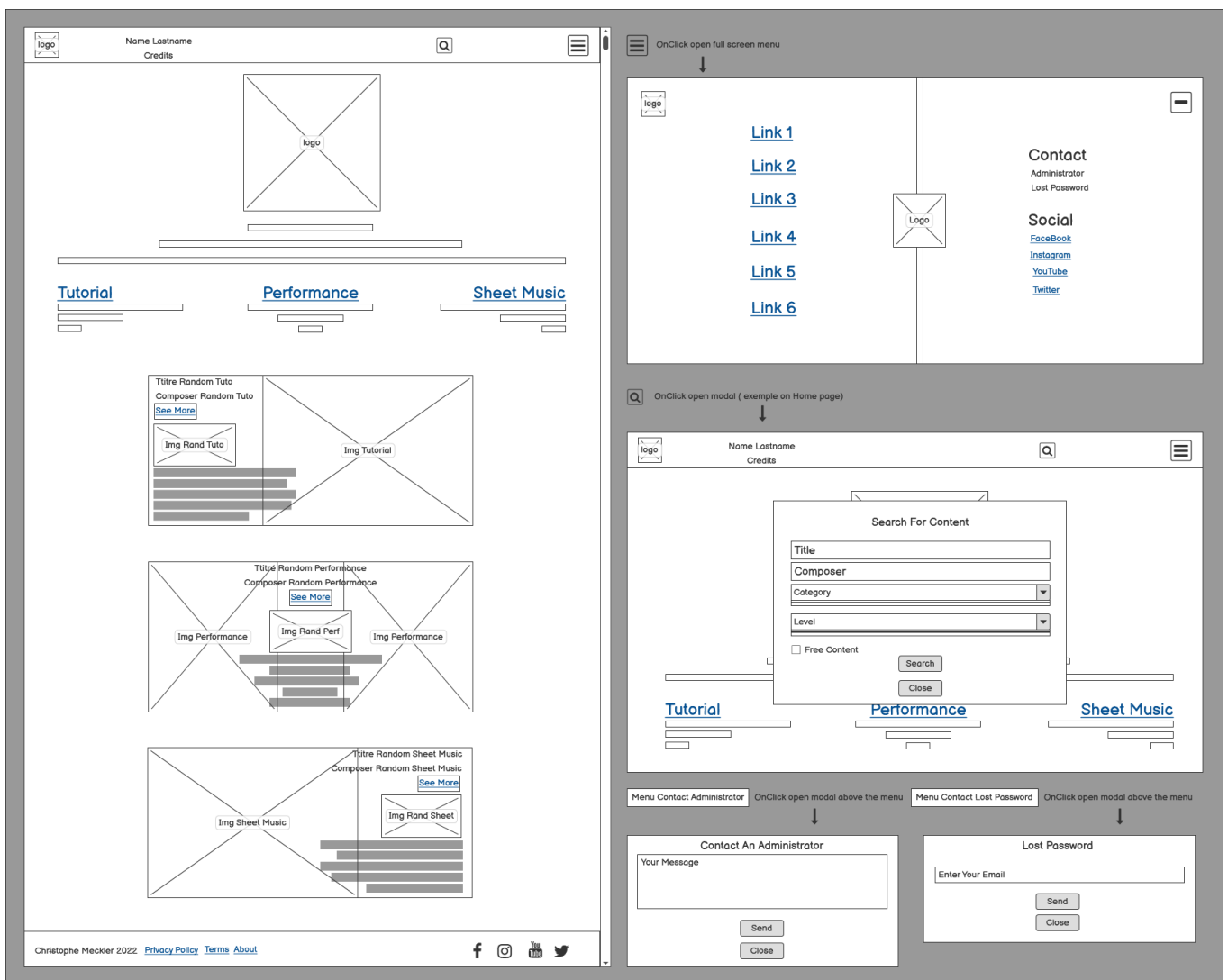
Intitulé
dossier_projet_meckler.pdf

ANNEXES

(Si le RC le prévoit)

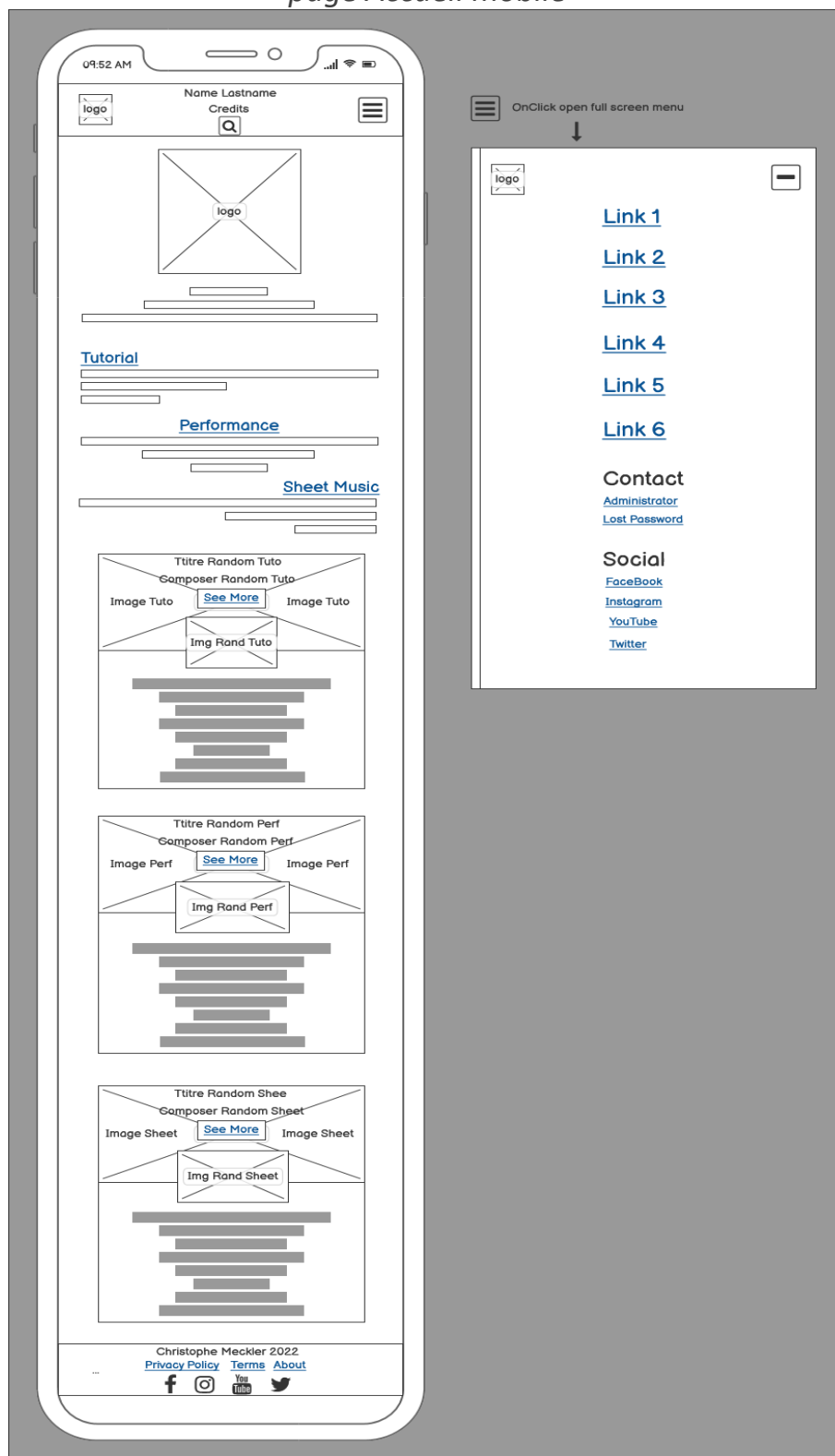
Annexe 1 : Wireframes

page Accueil desktop



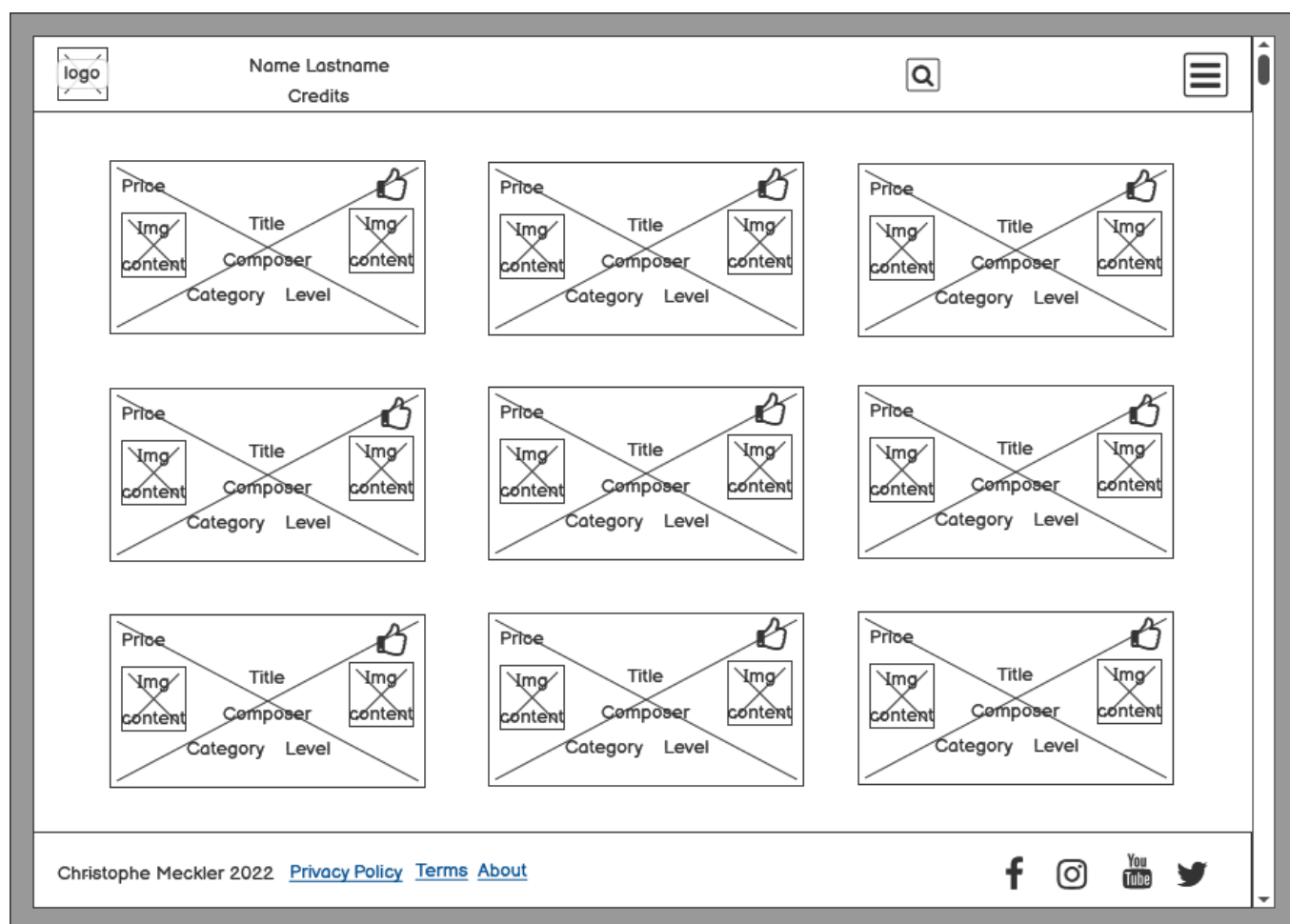
DOSSIER PROFESSIONNEL (DP)

page Accueil mobile



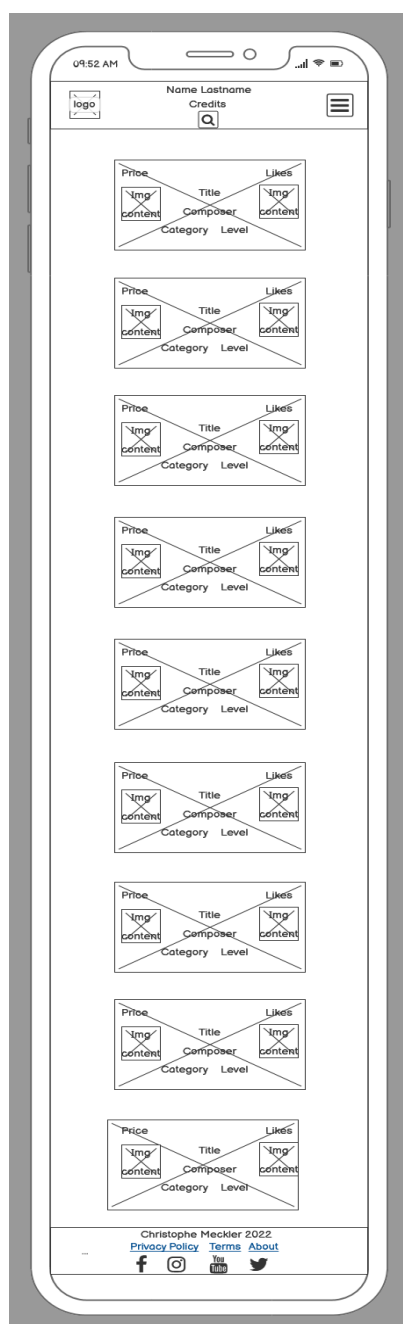
DOSSIER PROFESSIONNEL (DP)

page Contenu desktop



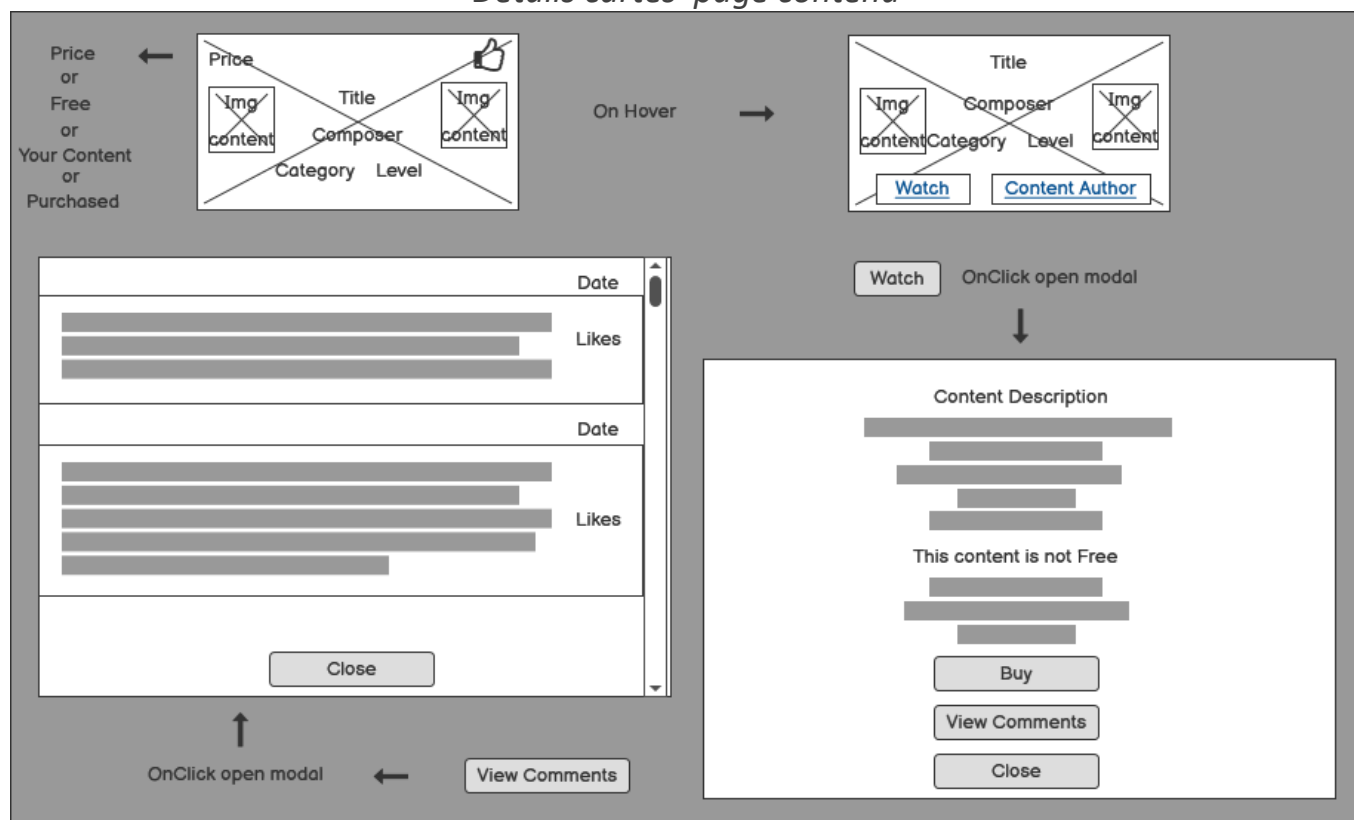
DOSSIER PROFESSIONNEL (DP)

page Contenu mobile






DOSSIER PROFESSIONNEL (DP)

Détails cartes page contenu







DOSSIER PROFESSIONNEL (DP)




page Inscription







Christophe Meckler 2022 [Privacy Policy](#) [Terms](#) [About](#)



page se connecter



Christophe Meckler 2022 [Privacy Policy](#) [Terms](#) [About](#)



DOSSIER PROFESSIONNEL (DP)

Page Mon compte

logo

Name Lastname

Credits

Email : email@email.com

Edit Your Name/LastName

Edit Your password

Add Content

Your Content

Purchased Content

Delete My Account

Notifications :

Date

Delete

Date

Delete

Christophe Meckler 2022

[Privacy Policy](#)

[Terms](#)

[About](#)

f

YouTube

Delete

or

Delete My Account

OnClick open modal →

Logo

Are You Sure ?

Delete

Close

Unfold →

Email : email@email.com

Edit Your Name/LastName

Name

Lastname

Add content

Edit Your password

Old Password

New Password

Confirm New Password

Add content

Add Content

Title

Composer

Description

Category

Level

Upload file

no files

Price

Add content

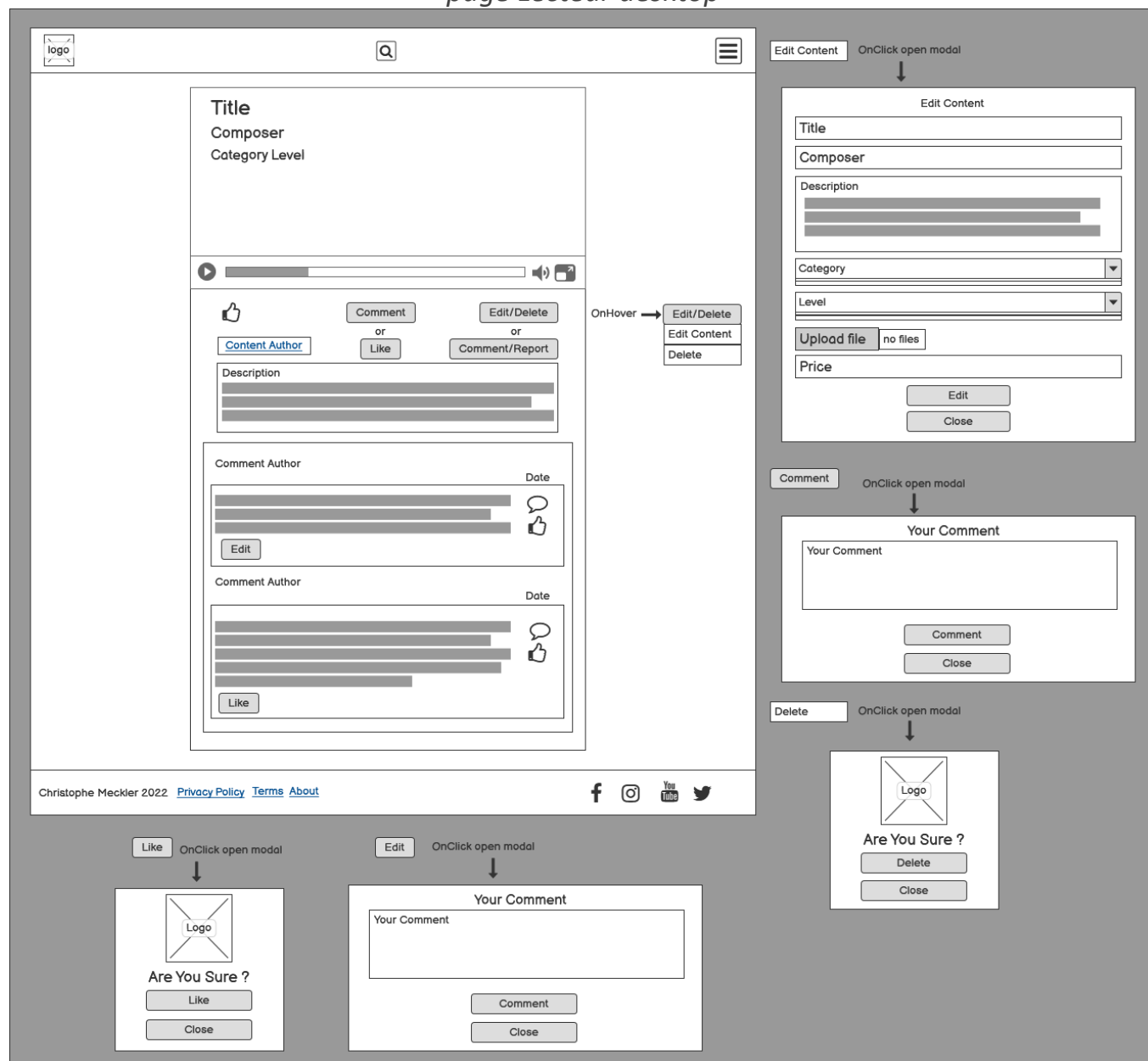
Your Content

Purchased Content

Delete My Account

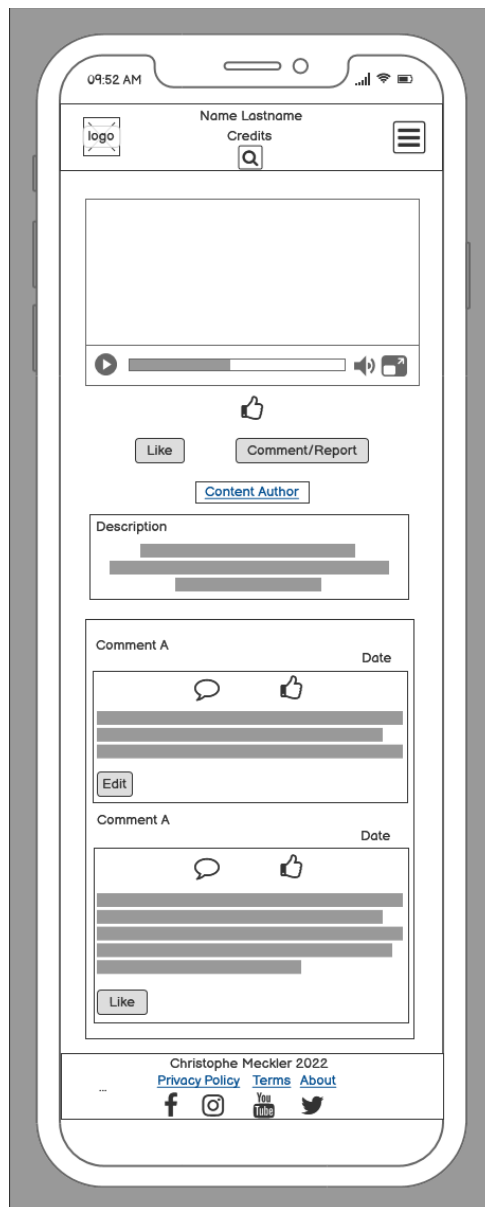
DOSSIER PROFESSIONNEL (DP)

page Lecteur desktop



DOSSIER PROFESSIONNEL (DP)

page Lecteur mobile



Annexe 2 : script SQL

```
-- phpMyAdmin SQL Dump
-- version 5.2.0
-- https://www.phpmyadmin.net/
--
-- Hôte : 127.0.0.1:3306
-- Généré le : Lun. 03 oct. 2022 à 14:08
-- Version du serveur : 5.7.36
-- Version de PHP : 8.1.10

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Base de données : `diplome`
--
```

DOSSIER PROFESSIONNEL (DP)

-- -----

--

-- *Structure de la table `comments`*

--

```
DROP TABLE IF EXISTS `comments`;  
  
CREATE TABLE IF NOT EXISTS `comments` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `comment` text NOT NULL,  
  `date` text NOT NULL,  
  `likes` int(11) NOT NULL DEFAULT '0',  
  `id_contents` int(11) NOT NULL,  
  `id_users` int(11) NOT NULL,  
  PRIMARY KEY (`id`),  
  KEY `comments_id_contents_FK` (`id_contents`),  
  KEY `comments_id_users_FK` (`id_users`)  
) ENGINE=InnoDB AUTO_INCREMENT=16 DEFAULT CHARSET=utf8;
```

-- -----

--

-- *Structure de la table `contact`*

--

```
DROP TABLE IF EXISTS `contact`;
```

DOSSIER PROFESSIONNEL (DP)

```
CREATE TABLE IF NOT EXISTS `contact` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `message` text NOT NULL,  
  `date` text NOT NULL,  
  `id_users` int(11) NOT NULL,  
  PRIMARY KEY (`id`),  
  KEY `contact_id_users_FK` (`id_users`)  
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8;
```

--

-- *Structure de la table `contents`*

--

```
DROP TABLE IF EXISTS `contents`;  
CREATE TABLE IF NOT EXISTS `contents` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `title` varchar(255) NOT NULL,  
  `content` varchar(255) NOT NULL,  
  `price` int(11) NOT NULL,  
  `composer` varchar(255) NOT NULL,  
  `category` varchar(255) NOT NULL,  
  `level` varchar(255) NOT NULL,  
  `description` text NOT NULL,
```


DOSSIER PROFESSIONNEL (DP)

```
`likes` int(11) NOT NULL DEFAULT '0',
`reporting` int(11) NOT NULL DEFAULT '0',
`id_users` int(11) NOT NULL,
PRIMARY KEY (`id`),
KEY `contents_id_users_FK` (`id_users`)
) ENGINE=InnoDB AUTO_INCREMENT=15 DEFAULT CHARSET=utf8;
```

-- -----

--

-- *Structure de la table `likes`*

--

```
DROP TABLE IF EXISTS `likes`;
CREATE TABLE IF NOT EXISTS `likes` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `id_comments` int(11) DEFAULT NULL,
  `id_contents` int(11) DEFAULT NULL,
  `id_users` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `likes_id_users_FK` (`id_users`),
  KEY `likes_id_comments_FK` (`id_comments`),
  KEY `likes_id_contents_FK` (`id_contents`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8;
```

-- -----

DOSSIER PROFESSIONNEL (DP)

--

-- Structure de la table `notifications`

--

```
DROP TABLE IF EXISTS `notifications`;  
CREATE TABLE IF NOT EXISTS `notifications` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `notification` text NOT NULL,  
  `date` text NOT NULL,  
  `id_users` int(11) NOT NULL,  
  PRIMARY KEY (`id`),  
  KEY `notifications_id_users_FK` (`id_users`)  
) ENGINE=InnoDB AUTO_INCREMENT=22 DEFAULT CHARSET=utf8;
```

-- -----

--

-- Structure de la table `purchased_contents`

--

```
DROP TABLE IF EXISTS `purchased_contents`;  
CREATE TABLE IF NOT EXISTS `purchased_contents` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `id_contents` int(11) NOT NULL,
```

DOSSIER PROFESSIONNEL (DP)

```
`id_users` int(11) NOT NULL,  
`original_price` int(11) NOT NULL,  
`buyer_repayment` int(11) NOT NULL DEFAULT '0',  
PRIMARY KEY (`id`),  
KEY `purchased_contents_id_contents_FK` (`id_contents`),  
KEY `purchased_contents_id_users_FK` (`id_users`)  
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8;
```

--

-- *Structure de la table `reporting`*

--

```
DROP TABLE IF EXISTS `reporting`;  
CREATE TABLE IF NOT EXISTS `reporting` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `message` text NOT NULL,  
  `date` text NOT NULL,  
  `id_users` int(11) NOT NULL,  
  `id_contents` int(11) NOT NULL,  
  PRIMARY KEY (`id`),  
  KEY `reporting_id_users_FK` (`id_users`),  
  KEY `reporting_id_contents_FK` (`id_contents`)  
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=utf8;
```

DOSSIER PROFESSIONNEL (DP)

--

-- Structure de la table `users`

--

```
DROP TABLE IF EXISTS `users`;  
  
CREATE TABLE IF NOT EXISTS `users` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `name` varchar(255) NOT NULL,  
  `lastname` varchar(255) NOT NULL,  
  `email` varchar(255) NOT NULL,  
  `password` varchar(255) NOT NULL,  
  `type` varchar(255) NOT NULL DEFAULT 'user',  
  `credits` int(11) NOT NULL DEFAULT '500',  
  `time` int(11) NOT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=39 DEFAULT CHARSET=utf8;
```

--

-- Contraintes pour les tables déchargées

--

--

-- Contraintes pour la table `comments`

DOSSIER PROFESSIONNEL (DP)

```
--  
  
ALTER TABLE `comments`  
  
    ADD CONSTRAINT `comments_id_contents_FK` FOREIGN KEY (`id_contents`) REFERENCES `contents`  
    (`id`) ON DELETE CASCADE,  
  
    ADD CONSTRAINT `comments_id_users_FK` FOREIGN KEY (`id_users`) REFERENCES `users` (`id`) ON  
    DELETE CASCADE;  
  
--  
  
-- Contraintes pour la table `contact`  
  
--  
  
ALTER TABLE `contact`  
  
    ADD CONSTRAINT `contact_id_users_FK` FOREIGN KEY (`id_users`) REFERENCES `users` (`id`) ON  
    DELETE CASCADE;  
  
--  
  
-- Contraintes pour la table `contents`  
  
--  
  
ALTER TABLE `contents`  
  
    ADD CONSTRAINT `contents_id_users_FK` FOREIGN KEY (`id_users`) REFERENCES `users` (`id`) ON  
    DELETE CASCADE;  
  
--  
  
-- Contraintes pour la table `likes`  
  
--  
  
ALTER TABLE `likes`  
  
    ADD CONSTRAINT `likes_id_comments_FK` FOREIGN KEY (`id_comments`) REFERENCES `comments`  
    (`id`) ON DELETE CASCADE,  
  
    ADD CONSTRAINT `likes_id_contents_FK` FOREIGN KEY (`id_contents`) REFERENCES `contents`
```

```
(`id`) ON DELETE CASCADE,

ADD CONSTRAINT `likes_id_users_FK` FOREIGN KEY (`id_users`) REFERENCES `users` (`id`) ON
DELETE CASCADE;

--

-- Contraintes pour la table `notifications`

--

ALTER TABLE `notifications`

ADD CONSTRAINT `notifications_id_users_FK` FOREIGN KEY (`id_users`) REFERENCES `users`
(`id`) ON DELETE CASCADE;

--

-- Contraintes pour la table `purchased_contents`

--

ALTER TABLE `purchased_contents`

ADD CONSTRAINT `purchased_contents_id_contents_FK` FOREIGN KEY (`id_contents`) REFERENCES
`contents` (`id`) ON DELETE CASCADE,

ADD CONSTRAINT `purchased_contents_id_users_FK` FOREIGN KEY (`id_users`) REFERENCES `users`
(`id`) ON DELETE CASCADE;

--

-- Contraintes pour la table `reporting`

--

ALTER TABLE `reporting`

ADD CONSTRAINT `reporting_id_contents_FK` FOREIGN KEY (`id_contents`) REFERENCES `contents`
(`id`) ON DELETE CASCADE,

ADD CONSTRAINT `reporting_id_users_FK` FOREIGN KEY (`id_users`) REFERENCES `users` (`id`)
ON DELETE CASCADE;
```

DOSSIER PROFESSIONNEL (DP)

COMMIT;

/!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;*

/!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;*

/!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;*