



DOSSIER PROFESSIONNEL (DP)

Nom de naissance ▶ FARKAS
Nom d'usage ▶
Prénom ▶ Denis
Adresse ▶ 35 Avenue des Borromées, bât M, 13012 Marseille

Titre professionnel visé

Développeur Web et Web Mobile

MODALITE D'ACCES :

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

DOSSIER PROFESSIONNEL ^(DP)

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.

DOSSIER PROFESSIONNEL ^(DP)

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

p. 5

► Maquetter une application

p. 5

► Réaliser une interface utilisateur web statique et adaptable

p. 7

► Développer une interface utilisateur web dynamique

p. 10

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

p. 12

► Créer une base de données

p. 12

► Développer les composants d'accès aux données

p. 17

► Elaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce

p. 21

Titres, diplômes, CQP, attestations de formation *(facultatif)*

p. 24

Déclaration sur l'honneur

p. 25

Documents illustrant la pratique professionnelle *(facultatif)*

p. 26

Annexes *(Si le RC le prévoit)*

p. 27

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 ► Maquetter une application

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

Nous avons tenté mes camarades et moi, dans le cadre du projet « social-network », de réaliser le design d'interface de notre application web. Pour ce faire nous avons appliqué une démarche de conception itérative passant successivement de simples schémas à la main (sketchs) à une zonification, puis des wireframes, mockups et enfin quelques prototypes.

Notre projet devait intégrer un ensemble de modules suggestions d'amis, groupes, suggestions de groupes, événements, suggestions d'événements, posts, ajout de commentaires, etc...

Chaque module pouvait se représenter sous la forme d'un bloc rectangulaire horizontal ou vertical selon sa disposition dans l'interface générale qui allait être séparée en trois colonnes, deux asides à gauche et à droite et une colonne centrale plus large.

Le fait de travailler sur des modules nous a permis de nous répartir le travail de manière équitable, chacun devant réaliser sur un compte moqps commun un ensemble de 6 wireframes.

Etant limités par la version gratuite de moqps, qui nous oblige à générer seulement 200 éléments, nous avons adopté une pratique agile : les éléments les plus fréquents (boutons, cercles d'avatars, icones, header des cards, zone de texte, footer de cards, etc.) étaient créés selon les besoins puis conservés dans la zone de travail collaborative. Chaque intervenant puisait dans ce fond commun pour réaliser son module, en copier-coller, prenait une capture de son module avec l'outil de capture de windows 10 et le conservait dans un répertoire sur un google drive partagé.

Disposant ainsi de plus de 18 wireframes de modules, incluant le header et footer de la page ainsi que les formulaires de connexion, inscription nous pouvions commencer à les intégrer dans les pages finales.

Pour réaliser cette opération nous avons sélectionné deux approches : l'insertion des blocs dans une image vierge à l'aide de photopad image editor ou l'insertion des images dans un document publisher. Cette dernière option nous permettait de grouper certains modules afin de les multiplier sur la page finale pour un rendu plus réel ou de recréer un élément du module en cas de besoin.

Nous disposions donc de maquettes filaires pour chaque page, sans design graphique, sans dimension esthétique et sans détails sur les éléments de l'interface utilisateur.

La seconde étape consistait à réaliser un prototype de nos pages en révélant l'identité et les fonctionnalités de base de l'interface finale pour un meilleur impact visuel.

DOSSIER PROFESSIONNEL ^(DP)

Ne disposant pas de temps pour réaliser chacun des éléments en design graphique, nous avons tenté d'utiliser uizard.io pour transformer nos wireframes en prototypes fonctionnels. S'agissant d'un nouveau service, soit disant entièrement automatisé, le rendu final n'a jamais atteint nos espérances.

Ayant une charte graphique pour cet exercice, inspiré d'un template de réseau social « sociala » et d'une librairie bootstrap modifiée dans le site bootswatch, nous avons réunis les éléments graphiques déjà prototypés dans ces ressources à l'aide de figma pour chacun des modules concernés dans les pages correspondantes.

Le résultat final n'est pas à proprement parler un prototype, il ne peut pas servir à la réalisation de tests utilisateurs, mais il s'en approche sous la forme d'un mockup en haute-fidélité. (Cf. : annexe 1-1-1)

2. Précisez les moyens utilisés :

Pour réaliser cette tâche nous avons utilisé :

- Microsoft Office Publisher
- Moqups.com
- PhotoPad Image Editor
- Figma
- Uizard.io

3. Avec qui avez-vous travaillé ?

Deux camarades, Thuc-Nhi Wiedenhofer et Emmanuel Cabassot qui participaient à ce projet, car nous devons réaliser une quinzaine de wireframes de manière collaborative.

4. Contexte

Nom de l'entreprise, organisme ou association ► *Coding school La plateforme*

Chantier, atelier, service ► *1^{er} année coding school*

Période d'exercice ► Du : *29/04/2021* au : *31/04/2021*

5. Informations complémentaires (facultatif)

Activité-type 1

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

Exemple n°2 ► Réaliser une interface utilisateur web statique et adaptable

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

Cet exemple provient d'un projet de réseau social réalisé cette année au sein de mon école la plateforme.

Il s'agit de la première page de notre site, une simple page de login qui va nous permettre de détailler les étapes nécessaires à sa réalisation. Dans sa version finale, cette page est dynamique, articulant du javascript avec une api en back-end en php. Nous n'allons traiter ici que son aspect statique.

Pour réaliser cette interface, avec versionning git, nous allons directement dans notre compte github.com pour créer un nouveau dépôt (repository) au nom de notre site. Disposant dans notre pc de gitkraken version pro, nous clonons ce site dans le répertoire www de notre environnement LAMP local de développement WAMP 64.

Nous ouvrons dans notre IDE Visual Code ce répertoire et nous créons à la racine du site un dossier assets contenant un dossier js pour les scripts js (site/assets/js), un dossier css pour les feuilles de style css (site/assets/css) et un dossier img pour les fichiers image (site/assets/img).

Dans css nous gardons un fichier bootstrap.css (version locale du framework bootstrap 4) une feuille de style style.css pour nos styles personnels.

Nous créons une page index.php dans notre IDE Visual code dans laquelle nous intégrons en premier lieu le type de document, la langue utilisée et tout un ensemble de propriétés que nous définissons dans la balise <head> : le charset (UTF-8 encodage Unicode pour un jeu de caractères universel), le viewport (sur la valeur spéciale device-width, qui est la largeur de l'écran en pixels CSS à une échelle de 100%), le titre, des méta balises qui permettent d'indiquer des informations pour le référencement des moteurs de recherche, le chargement des feuilles de style (bootstrap et style pour nos apports CSS personnels). (Cf. annexe 1-2-1)

Puis nous transposons en code, à l'aide de balises html 5 dans la partie <body> les différentes parties et sections créées lors du maquettage, en respectant l'ordre de présentation correspondant à <header>, <main> et <footer>.

La bonne pratique consiste à indenter son code pour que chaque balise ouvrante commence à la tabulation suivante, de telle manière que leur balise fermante soit au même niveau. Cela nous permet de ne pas oublier de fermer les balises qui le nécessitent, d'un simple regard.

L'utilisation du framework Bootstrap 4, de son modèle de grille en flexbox pour notre feuille de style CSS détermine le caractère « responsive » de la page (sa capacité à s'adapter à différentes tailles d'écran).

Nous divisons chaque sous-section de la page, considérée alors comme une ligne (row), en une ou plusieurs colonnes qui utilisent une partie des 12 fractions égales de la page sur un axe horizontal.

Dans notre exemple pour le formulaire de login dans la section `<main class="container mt-5">` nous définissons une `<div class="row ">`, donc une ligne, qui va afficher trois colonnes côte à côte `<div class="col-xl-2"></div> <div class="col-xl-4 col-sm-12"></div> <div class="col-xl-6 col-sm-12"></div>`

Le total de ces trois colonnes en xl, c'est-à-dire pour bootstrap en format large d'écran, supérieur à 1200px est de 2+4+6 soit 12 divisions. Lorsque nous regardons cette page sur un mobile (entre 576px et 767px) la classe bootstrap col-sm-12 indique que chacune de ces deux colonnes occupera la largeur totale de l'axe horizontal, soit les 12 divisions. La troisième colonne apparaîtra donc en dessous de la seconde.

La seconde colonne contient une balise image qui représente l'image d'accueil d'instagram. ``. La propriété html **src** permet de spécifier le chemin du répertoire qui contient cette image ainsi que le nom - extension de l'image.

Basculant sur une colonne de 12 divisions en format mobile, nous n'appliquons pas dans ce cas la classe img-responsive de bootstrap à cette image.

La troisième colonne contient par contre une image à laquelle nous appliquons une classe w-50 soit la propriété en CSS `width=50%` de son parent, ici la div col. Il s'agit d'un moyen simple d'appliquer un rapport largeur d'image par rapport à la colonne qui elle-même dépendra de la taille de l'écran.

Cette colonne contient aussi un formulaire `<form>` qui sera traité par la suite par JavaScript donc pas de propriété `action` ou `method` dans ce cas.

Chaque champ input est intégré dans une div avec la classe `form-group` de bootstrap qui applique une marge de 1rem au pied (bottom). Pour chacun d'eux nous appliquons la classe `form-control` qui correspond dans bootstrap à plus de 63 cas de figures déterminant les comportements, les dimensions ainsi que la couleur des champs selon le type d'action (focus, etc...). Ces comportements sont déjà configurés dans `bootstrap.min.js`.

Nous fermons ensuite en respectant l'indentation le `</form>`, sa colonne `</div>`, la ligne contenant les trois colonnes `</div>` et enfin la section `</main>`.

Nous plaçons la section footer, qui dans cet exemple ne contient pas d'élément particulier.

Puis enfin viennent les balises `<script></script>` qui indiquent dans la propriété **src** le chemin de la bibliothèque `jquery.min.js`, de la bibliothèque `bootstrap.bundle.min.js` ainsi que le chemin/nom du fichier JavaScript que nous allons utiliser.

A chaque étape de création de code nous pouvons visualiser dans notre explorateur à l'adresse `localhost/nom du site`, donc dans le serveur apache local de Wamp64, le résultat. Et cela parce que par convention pour la suite de notre projet, nous avons donné à notre page l'extension `.php` et qu'elle nécessite donc d'être générée par le serveur avant d'être affichée par notre explorateur.

En cas de problème d'affichage, un simple clic droit sur notre page, affichée dans notre explorateur google chrome, permet d'accéder à la fonction Inspection de celui-ci. Nous accédons donc à l'ensemble du code html pour vérifier que les balises soient bien fermées, au bon endroit pour respecter la structure de nos sections, ainsi que les propriétés CSS de chaque élément sélectionné, pour vérifier que les propriétés définies dans notre feuille de style soient bien présentes au moment de l'affichage.

DOSSIER PROFESSIONNEL ^(DP)

2. Précisez les moyens utilisés :

Pour réaliser cette tâche nous avons utilisé :

- Github. com pour le versionning
- GitKraken pro pour effectuer les commandes git en interface
- Visual code pour l'ide (environnement de développement « intégré »)
- Wamp64 pour un environnement LAMP local
- Google chrome pour la visualisation et l'inspection
- Bootstrap 4 pour le framework CSS
- HTML 5 pour le code des balise html

3. Avec qui avez-vous travaillé ?

Pour cette page, j'étais seul

4. Contexte

Nom de l'entreprise, organisme ou association ► *Coding school La plateforme*

Chantier, atelier, service ► *1^{er} année coding school*

Période d'exercice ► Du : *01/05/2021* au : *02/05/2021*

5. Informations complémentaires (facultatif)

Activité-type 1

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

Exemple n° 3 ► Développer une interface utilisateur web dynamique

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

Cet exemple reprend notre page de connexion index.php du premier exemple de l'activité-type 1, afin d'appliquer un script JavaScript sur le formulaire, pour : vérifier que les champs du formulaire soient bien remplis, récupérer les valeurs introduites par l'utilisateur et les diriger vers notre API php en back-end à l'aide d'Ajax.

Nous ouvrons dans notre IDE Visual Code ce répertoire et nous créons dans le dossier assets/js qui contient déjà la librairie JQuery, un script que nous avons appelé login.js. (Cf. annexe 1.3.1)

Ce script commence par identifier le formulaire dans le document "document.getElementById" grâce à une id "formSignIn" que nous avons placé dans la balise html <form> pour en faire une variable nommée formL.

Nous plaçons sur le document une écoute "addEventListener" qui se déclenche lorsque le DOM de la page est entièrement chargé "DOMContentLoaded". Cette écoute prépare une autre écoute sous forme de fonction sur le formulaire qui se déclenchera lorsque "submit" sera activé. Cela appellera la fonction "login()".

Pour vérifier que les champs du formulaire soient bien remplis nous avons utilisé l'attribut "required" d'html5 directement dans les balises <input>, car il dispose aujourd'hui d'une compatibilité plus large :

Navigateurs	Versions
Firefox	Firefox 4.0+
Opera	Opera 9.5+
Google Chrome	Chrome 3.0+
Internet Explorer	Internet Explorer 10+

Les contrôles de sécurité sur les valeurs introduites dans nos inputs seront directement appliqués dans notre API php en back-end.

Ces valeurs introduites sont récupérées par une fonction "serializeForm" qui utilise le "FormData" des spécifications XMLHttpRequest en créant un nouvel objet "Formdata" pour récupérer les keys. "Formdata" est généralement compatible sauf pour internet explorer 10. La condition étant que chaque balise dispose d'un attribut name.

La fonction "login()" convertit cet objet JavaScript en JSON chaîne de caractères avec "JSON.stringify()" et crée un objet "XMLHttpRequest" (XHR) permettant d'interagir avec notre API php dans une approche AJAX.

Cet objet envoie à notre contrôleur dans l'api en post le JSON obtenu ainsi que les headers "content-type" et "text/plain".

DOSSIER PROFESSIONNEL ^(DP)

Il attend une réponse en écoutant le "readystatechange" de "xhr". Si le "readystate" est strictement égal à 4 et le code du statut de la réponse est 200, c'est à dire si l'email se trouve bien dans notre base de données dans la table user et le password correspond bien, notre script redirige l'utilisateur vers la page home.

Dans le cas contraire, il introduit à l'aide JQuery un message sous forme de paragraphe <p></p> dans la balise html qui a l'id "resultat".

Nous pouvons vérifier le fonctionnement de notre script en inspectant notre page login.php dans google Chrome, au niveau de la console et du réseau XHR. Nous accédons ainsi aux valeurs envoyées, aux headers et à la réponse retournée. Nous réservons le contrôle des requêtes et de notre API en utilisant POSTMAN pour afficher les messages d'erreur SQL et PHP.

2. Précisez les moyens utilisés :

Pour réaliser cette tâche nous avons utilisé :

- Github.com pour le versionning
- GitKraken pro pour effectuer les commandes git en interface
- Visual code pour l'ide (environnement de développement « intégré »)
- Wamp64 pour un environnement LAMP local
- Google chrome pour la visualisation et l'inspection
- HTML 5 pour le code des balise html et le required
- La librairie JQuery
- Le framework bootstrap 4

3. Avec qui avez-vous travaillé ?

Sur cette page, seul

4. Contexte

Nom de l'entreprise, organisme ou association ► *Coding school La plateforme*

Chantier, atelier, service ► *1^{er} année coding school*

Période d'exercice ► Du : *03/05/2021* au : *04/05/2021*

5. Informations complémentaires (facultatif)

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 d'un exercice de boutique online nous devons construire une base de données pour enregistrer ses articles, les afficher de manière dynamique, établir la procédure d'achat, de paiement, de livraison et de facturation pour le client, ainsi que d'autres enregistrement comme l'adresse du client, etc...

La première étape consistait à conceptualiser une application qui permette de représenter la vente de chapeaux panamas.

Pour ce faire, nous avons réalisé à l'aide de Looping le modèle conceptuel de données de notre application en mettant en évidence les entités et les associations.

L'entité est un objet que l'on souhaite modéliser, par exemple une catégorie de produit, une gamme de couleurs ou de taille, un article, un client, son adresse, etc...

Chaque entité possède des attributs qui peuvent être standardisés par type (date, nombre entier, texte, chaîne de caractères, booléen) pour les décrire.

Les associations illustrent le lien entre les entités : les relations sémantiques. Par exemple, un article a une couleur, une taille, correspond à telle catégorie de produit qui possède tel attribut de calibre de palme, tel texte de description, etc.

On parle alors de cardinalité : le nombre de fois où une entité peut appartenir à une association.

Dans notre exemple nous avons identifié les entités suivantes :

User comme client,

Commande comme panier,

Détail commande comme élément du panier,

Article

Couleur

Taille

Catégorie Produit de l'article

Attributs du produit

Adresse de facturation ou de livraison

Mode de livraison

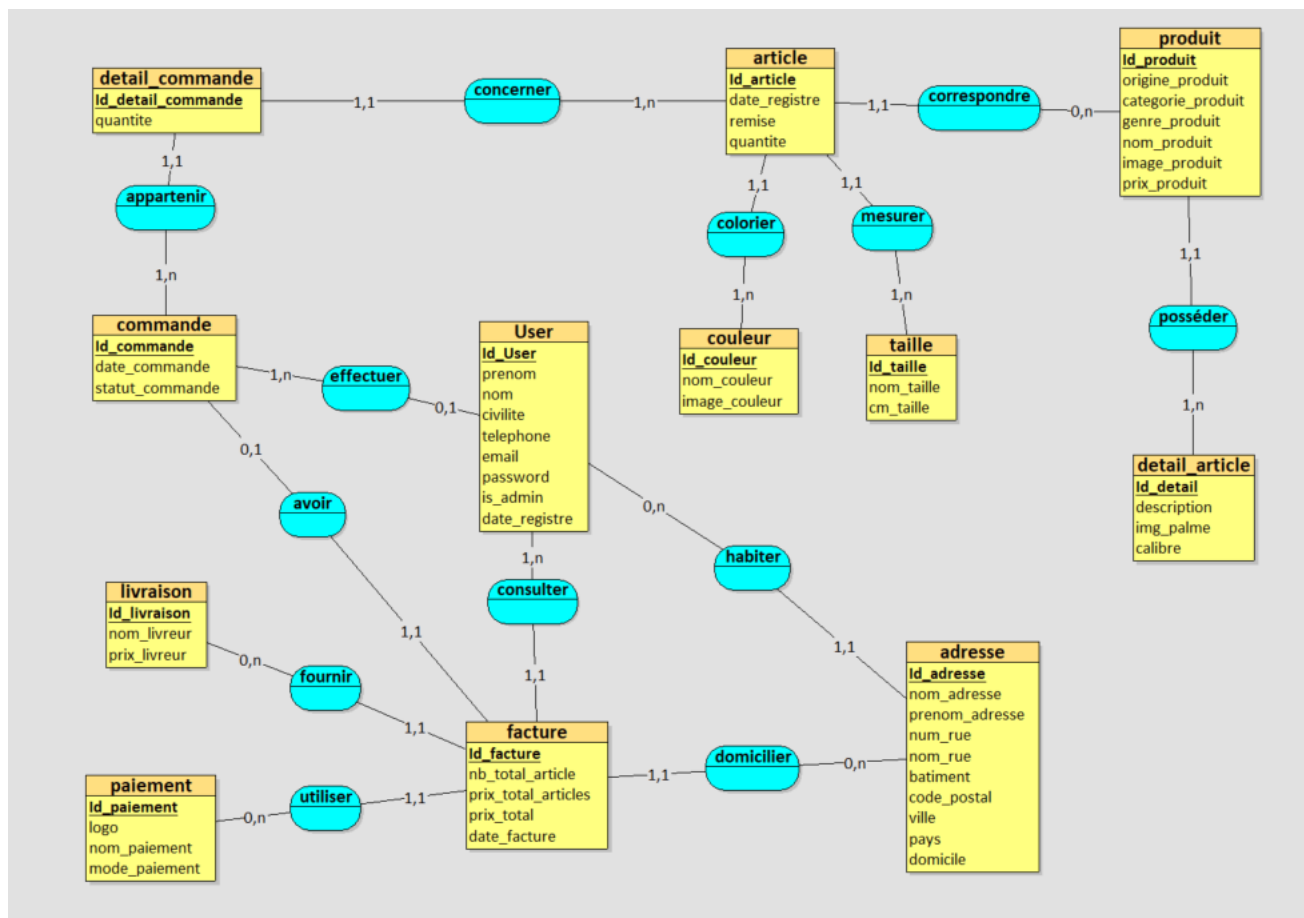
Mode de paiement

Facture qui récapitule tous les éléments de la commande du client et de livraison ainsi que du

DOSSIER PROFESSIONNEL (DP)

paiement.

Les associations et leur cardinalité sont représentées dans le MCD de la manière suivante :

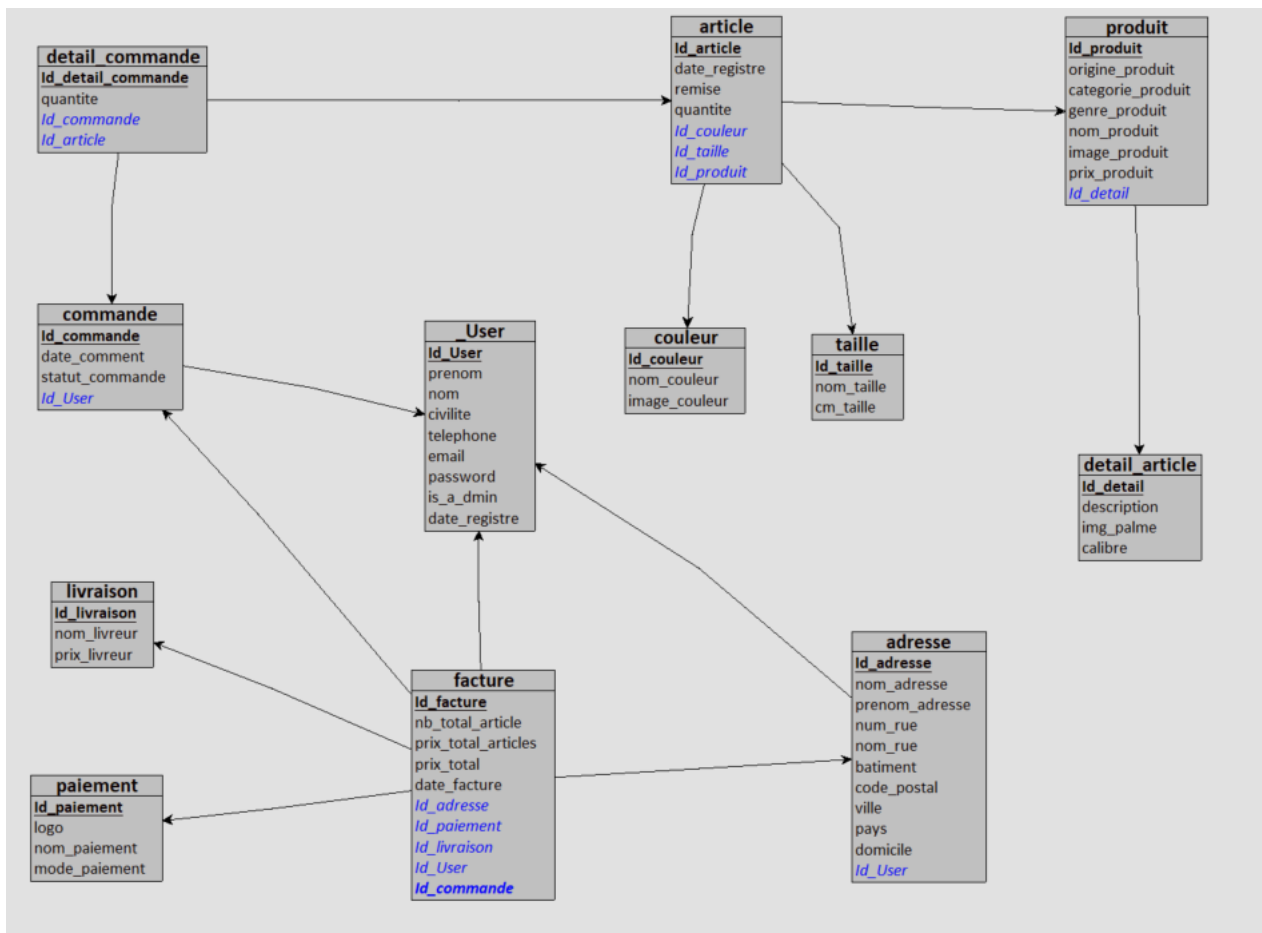


Looping nous permet par la suite de transformer ce MCD en modèle logique de données c'est-à-dire en un ensemble compréhensible pour un SGBD. Les entités sont remplacées par des éléments de bases de données : les tables.

A partir du MCD le MLD interprète les cardinalités pour comprendre quelles sont les clefs qui deviendront étrangères dans les tables en relation. Une cardinalité 1,n de facture vers paiement sera interprété comme l'ajout de la clef id_paiement dans la table facture comme clé étrangère.

Dans le cas où nous aurions une relation n,n le modèle logique ajouterait alors une table de transition qui contiendrait l'id de la première et de la seconde table.

DOSSIER PROFESSIONNEL (DP)



Looping nous permet aussi de générer le fichier SQL pour construire la base de données. (Cf.annexe 2.1.1)

Pour construire le modèle physique de données de cette application nous allons utiliser le mode reversing engineer de mysql workbench.

Nous importons tout d'abord le fichier sql dans une nouvelle table boutique créée dans php my admin. Les tables sont ainsi créées.

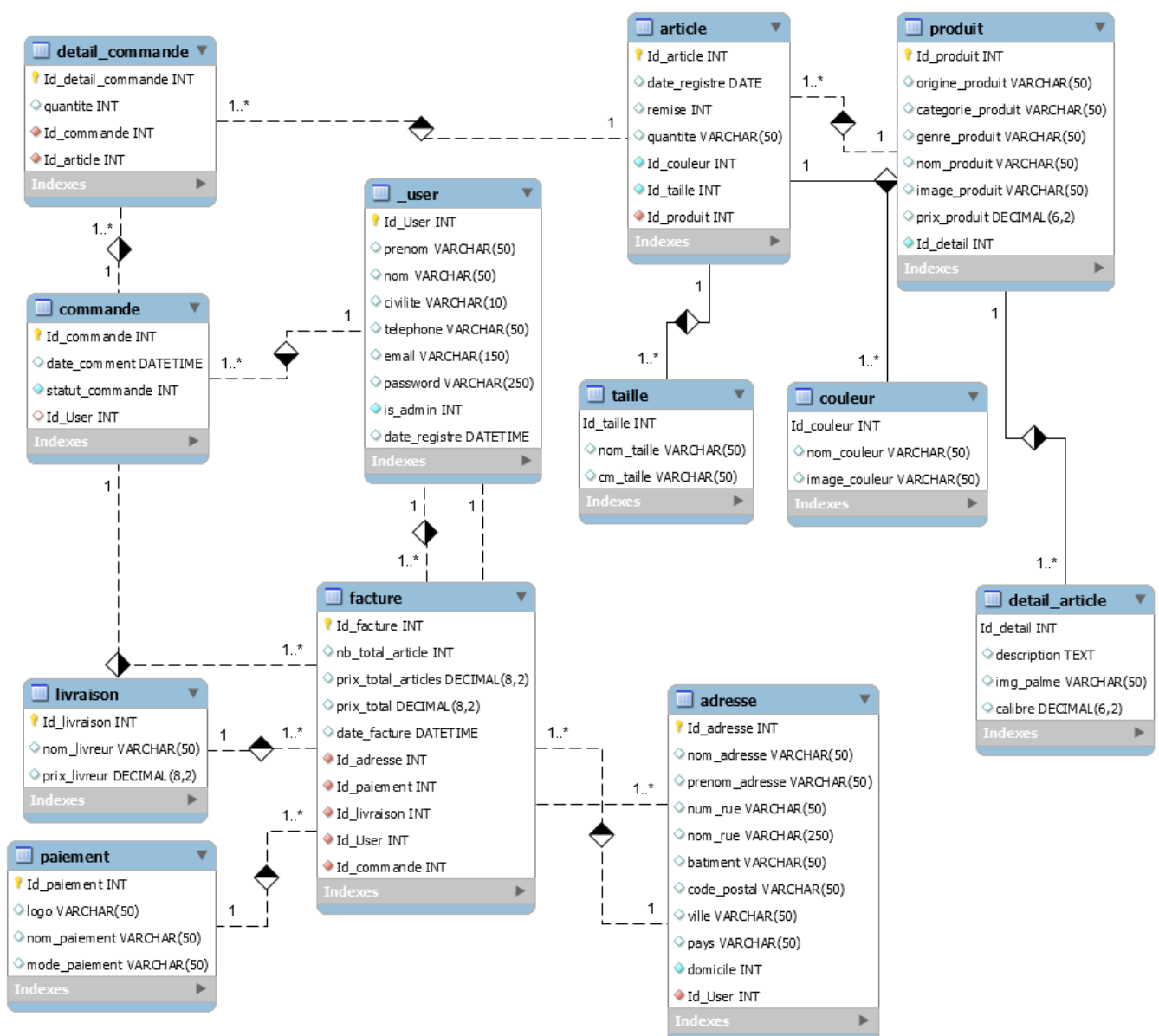
Mysql workbench a été configuré pour se connecter directement à cet environnement local. Dans le mode database de ERR nous sélectionnons l'assistant reverse engineer.

Il va récupérer la configuration de cette base et la traduire dans un diagramme de tables complet. Il nous suffit alors avec la fonction "cardinalité éprouvette" de retrouver les associations id - clef id

DOSSIER PROFESSIONNEL (DP)

étrangère entre les deux tables.

Voici le résultat final du modèle physique de données :



Nous avons ainsi pu vérifier la cohérence de nos entités, des associations, de leur cardinalité.

DOSSIER PROFESSIONNEL ^(DP)

La dernière étape consiste à insérer dans les tables livraison, paiement, detail_article, produit les informations qui restent immuables. La table article est remplie à ce moment mais les quantités seront modifiées automatiquement en fonction des achats et des ventes de la boutique.

Ces articles pour un souci de cohérence ne seront jamais éliminés. En cas de besoin pour un nouveau stock il suffira d'ajouter un article identique avec une nouvelle date de création, ce qui nous permet de gérer les stocks et des remises sur les articles trop anciens, le chapeau panama ayant une date de péremption.

2. Précisez les moyens utilisés :

Pour le modèle conceptuel de données et le modèle logique : Looping

Pour le modèle physique nous avons utilisé MySQL Worbench

Pour créer la Bdd : Php MyAdmin dans Wamp64

3. Avec qui avez-vous travaillé ?

Avec ma camarade de projet THuc-Nhi Wiedenhofer, nous avons défini les entités et les associations.

4. Contexte

Nom de l'entreprise, organisme ou association ► *Coding school La plateforme*

Chantier, atelier, service ► *1^{er} année coding school*

Période d'exercice ► Du : *01/04/2021* au : *04/04/2021*

5. Informations complémentaires (facultatif)

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 ce même cadre de projet boutique, nous avons construit une architecture MVC (modèle, vue, contrôleur) simple pour développer notre application. Nous allons détailler les composants d'accès aux données dans cet exemple pour expliquer leur fonctionnement.

Ce projet utilise la programmation orientée objet (POO) et le Php Data Object (PDO) qui constitue une couche d'abstraction qui intervient entre l'application PHP et un système de gestion de base de données (SGDB) tel que MySQL, PostgreSQL ou MariaDB par exemple.

Le fichier index.php à la racine de notre MVC, « autoload » une librairie database.php, de fait une classe pdo qui configure les éléments de connexion à la base de données. Les paramètres indiqués au constructeur sont la source de la base de données ainsi que les identifiants.

```
//constructeur PDO, qui recevra les valeurs de config.php
public function __construct() {
    $conn = 'mysql:host=' . $this->dbHost . ';dbname=' . $this->dbName;
    $options = array(
        PDO::ATTR_PERSISTENT => true,
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION
    );
    try {
        $this->dbHandler = new PDO($conn, $this->dbUser, $this->dbPass, $options);
    } catch (PDOException $e) {
        $this->error = $e->getMessage();
        echo $this->error;
    }
}
```

Cette classe définit aussi une méthode générique de requêtes, une autre de bind parameters selon le type de valeur (int, booléen, null et string) ainsi que des méthodes (fonctions publiques) pour appliquer un retour d'exécution simple, une méthode de fetchall, de fetch ou de rowCount. Selon le jeu de données attendu en résultat de nos requêtes (simple ligne ou multiples, par exemple) nous choisissons une de ces méthodes.

Dans le répertoire models de notre MVC, chaque model possède un constructeur quiinstanciera cette classe database.

```
class Achat {
    private $db;
    public function __construct() {
        $this->db = new Database;
    }
}
```

Le model définit aussi des fonctions publiques qui définissent les requêtes et le traitement de leur résultat à retourner.

Par exemple pour la lecture :

```
public function livraisonFacture($id_livraison){
    $this->db->query('SELECT * FROM livraison WHERE id_livraison= :id_livraison');
    $this->db->bind(':id_livraison', $id_livraison);
    $livraison = $this->db->single();
    return $livraison;
}
```

Cette requête a pour fonction de sélectionner dans la table livraison toutes les informations relatives à une id_livraison passée en paramètre de fonction. Elle prend la requête sous forme de SQL, puis traite avec « bind » la variable en paramètre afin d'éviter l'injection de code malveillant dans notre base de données.

Elle applique finalement au résultat de requête la fonction single() qui a été définie dans database.php comme une fonction execute suivie d'un fetch(PDO ::FETCH_OBJ) du statement définit comme :

```
public function query($sql) {
    $this->statement = $this->dbHandler->prepare($sql);
}
```

Le controller respectif à ce model contient un constructeur qui prend en héritage les attributs d'une autre classe générique appelée controller qui se charge d'instancier le nouveau model à l'aide d'une fonction publique model(\$model).

```
class Achats extends Controller {

    public function __construct() {

        $this->achatModel = $this->model('Achat');
```

La classe générique controller:

DOSSIER PROFESSIONNEL (DP)

```
class Controller {
    public function model($model) {
        //Requiert le fichier model
        require_once (ROOT.'models/' . $model . '.php');
        //Instancie model
        return new $model();
    }
}
```

En résumé, lorsque nous appelons dans notre MVC une adresse controller/method/parameter , ce controller est instancié par notre autoloader, ce qui par héritage de la classe générique controller instancie le model, ce qui instancie une nouvelle classe db de la classe database.

Nous disposons donc à ce moment d'une nouvelle connexion PDO à la base de données, ainsi que de toutes les méthodes référentes à database.php et au model.

Si le controller achats définit la variable \$livraison comme étant égale à \$this->achatModel->listLivraisons() ; il demande en fait au model de lancer une requête sql à la base de donnée définie dans la fonction publique

```
public function listLivraisons() {
    $this->db->query('SELECT * FROM livraison ');

    $livraisons = $this->db->resultSet();
    return $livraisons;
}
```

Qui appliquera la méthode query puis la méthode resultSet définies dans database.

Cette dernière retournera un objet résultat du fetchAll(PDO ::FETCH_OBJ) dont les noms de propriétés correspondent aux noms des colonnes.

Enfin pour lire cet objet dans une view il suffit de passer l'objet \$livraison dans un \$data, c'est-à-dire \$data = ['livraison' => \$livraison] ; puis d'envoyer ce \$data comme paramètre lorsque l'on convoque la view \$this->view(recapitulatif, \$data) ; comme le définit le controller général.

Pour afficher les attributs de \$livraison dans la vue nous faisons un

```
foreach($data['livraisons'] as $livreur)
```

puis un echo \$livreur->nom_livreur par exemple.

Avec cette architecture MVC nous pouvons effectuer toutes les requêtes du CRUD, create, read, update et delete sur les tables présentes dans notre base de données.

2. Précisez les moyens utilisés :

Pour réaliser cette tâche nous avons utilisé :

DOSSIER PROFESSIONNEL ^(DP)

- Github. com pour le versionning
- GitKraken pro pour effectuer les commandes git en interface
- Visual code pour l'ide (environnement de développement « intégré »)
- Wamp64 pour un environnement LAMP local
- Google chrome pour la visualisation et l'inspection
- La classe PDO
- La programmation orientée objet POO
- Une architecture de type MVC
- phpMyAdmin comme interface de gestion de base de données

3. Avec qui avez-vous travaillé ?

Pour cet exercice j'étais seul

4. Contexte

Nom de l'entreprise, organisme ou association ► *Coding school La plateforme*

Chantier, atelier, service ► *1^{er} année coding school*

Période d'exercice ► Du : *25/04/2021* au : *26/04/2021*

5. Informations complémentaires (facultatif)

Activité-type 2

Exemple n° 3 ► Développer la partie back-end d'une application web ou web mobile

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

Cet exemple reprend le projet boutique online, consacré à la vente de chapeaux panama avec une architecture de type MVC. Nous avons détaillé dans l'exemple précédent les composants d'accès aux données avec les Models. Nous allons décrire l'ensemble de la procédure Model View Controller pour réaliser un CRUD administrateur en back-office.

L'administrateur accède à cet espace en utilisant la même interface de connexion que les clients, donc nous vérifions son email et son password ; mais son enregistrement dans la table user contient dans le champ is_admin la valeur 1 pour oui et nous contrôlons aussi cette valeur.

Avec la connexion assurée, la nav bar au sommet de notre page indique : « vous êtes connecté comme Administrateur », affiche un link gestion et un link déconnexion pour terminer la session et retourner à home en étant déconnecté. (Cf. annexe 5-2-3)

En cliquant sur Gestion l'administrateur accède à la première page crudArticles.php. Cette view est le résultat d'une public function crudArticles() dans notre controller Admins :

```
public function crudArticles(){
    if (!empty($_SESSION['id_user']) && $_SESSION['is_admin']==1){
        $articles = $this->adminModel->crudArticles();

        $data = [
            'articles' => $articles
        ];

        $this->view('admins/crudArticles', $data);
    } else {
        header('location:' . WWW_ROOT . 'pages/index');
    }
}
```

Nous contrôlons avec une condition if si le \$_SESSION['id_user'] n'est pas vide et si \$_SESSION['is_admin'] est bien égal à 1 par mesure de sécurité.

Nous appelons la fonction crudArticles() de notre model.

DOSSIER PROFESSIONNEL (DP)

```
public function crudArticles(){
    $this->db-
>query('SELECT id_article, origine_produit, categorie_produit, genre_produit, nom_produ
it, nom_taille, nom_couleur, image_produit, date_registre, prix_produit, quantite, remi
se FROM article JOIN produit ON article.id_produit = produit.id_produit JOIN taille ON
article.id_taille = taille.id_taille JOIN couleur ON article.id_couleur = couleur.id_c
ouleur ');
    $articles=$this->db->resultSet();
    return $articles;
}
```

Cette fonction lance une requête SQL dans notre BDD pour récupérer toutes les caractéristiques de chaque article en joignant les tables article, produit, taille, couleur.

Le résultat de cette requête est traité par la méthode resultSet() c'est-à-dire un fetchAll(PDO ::FETCH_OBJ) sur statement et on retourne l'objet \$articles.

Il est placé dans \$data['articles'] et nous appelons la vue crudArticles avec en paramètre \$data

Cette vue contient une table pour chaque catégorie de chapeau comprenant 9 colonnes que nous alimentons en Php avec un foreach sur \$data['articles'] as \$article, plus une colonne avec un link modifier qui nous redirige vers une méthode controller formArticle avec pour paramètre l'id_article:

```
echo '<td><a href="'.WWW_ROOT .'admins/formArticle/'. $article->id_article.'">
    Modifier</a></td>';
echo "</tr>";
```

En cliquant sur modifier nous accédons au formulaire de l'article, ou chaque input a pour value, une variable obtenue par la fonction formArticle(\$id_article) de notre controller, qui lance la méthode viewArticle(\$id_article) de notre adminModel qui retourne un \$data['article'] propre à cet article.

Ce formulaire a pour action la fonction updateArticle() dans notre controller qui nous permet de modifier la quantité et la remise grâce à une requête SQL UPDATE dans la fonction updateArticle de notre model. Cette fonction dispose d'une mesure de sécurité supplémentaire :

```
if($_SERVER['REQUEST_METHOD'] == 'POST' && isset($_POST['update'])){
```

qui nous permet de contrôler dans la variable super globale \$_SERVER s'il s'agit bien d'un POST et si notre variable de contrôle \$_POST du formulaire est bien update.

Les autres informations appartiennent à la table produit et seront modifiables dans le CRUD produit.

Une fois réalisé le UPDATE notre controller nous redirige vers le CRUD articles en appliquant de nouveau la méthode crudArticles() de notre controller Admins, donc en chargeant les modifications.

DOSSIER PROFESSIONNEL ^(DP)

2. Précisez les moyens utilisés :

Pour réaliser cette tâche nous avons utilisé :

- Github.com pour le versionning
- GitKraken pro pour effectuer les commandes git en interface
- Visual code pour l'ide (environnement de développement « intégré »)
- Wamp64 pour un environnement LAMP local
- Google chrome pour la visualisation et l'inspection
- HTML 5 pour le code des balise html et le required
- Le framework bootstrap 4
- POO et PDO
- MVC simple avec autoloader sur index

3. Avec qui avez-vous travaillé ?

Sur cette page, seul

4. Contexte

Nom de l'entreprise, organisme ou association ► *Coding school La plateforme*

Chantier, atelier, service ► *1^{er} année coding school*

Période d'exercice ► Du : *12/04/2021* au : *13/04/2021*

5. Informations complémentaires (facultatif)

DOSSIER PROFESSIONNEL ^(DP)

Titres, diplômes, CQP, attestations de formation

(facultatif)

Intitulé	Autorité ou organisme	Date
DEUG sciences Economiques	AIX Marseille II	1986
Maîtrise Ethnologie	AIX MARSEILLE I	1992
DEA Anthropologie	IHEAL PARIS III nouvelle Sorbonne	1994
6ième année Doctorat anthropologie	IHEAL PARIS III nouvelle Sorbonne	2000

DOSSIER PROFESSIONNEL ^(DP)

Déclaration sur l'honneur

Je soussigné(e) DENIS FARKAS ,

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 MARSEILLE,

le 20/06/2021

pour faire valoir ce que de droit.

Signature :

DOSSIER PROFESSIONNEL ^(DP)

Documents illustrant la pratique professionnelle

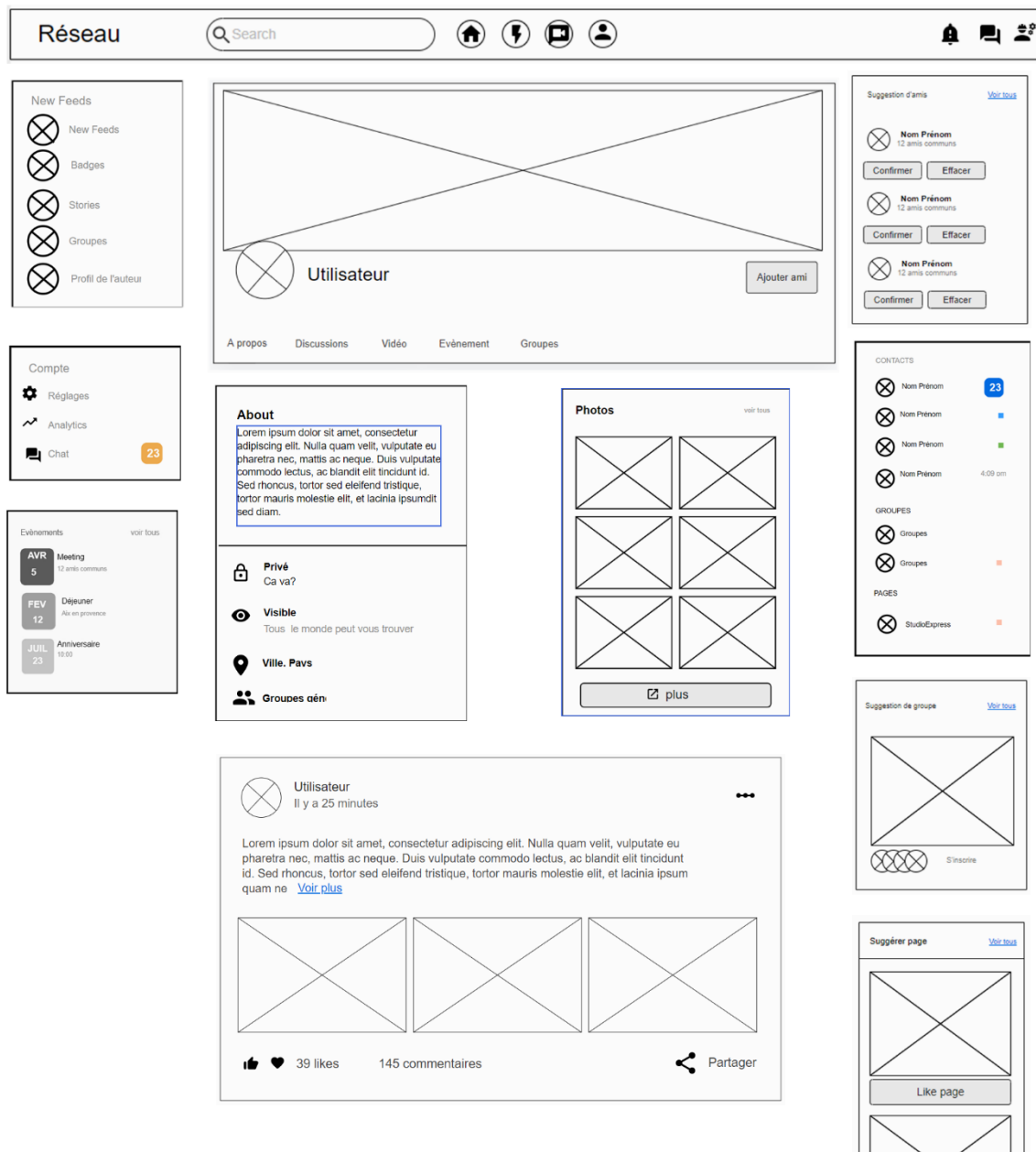
(facultatif)

Intitulé
Cliquez ici pour taper du texte.

DOSSIER PROFESSIONNEL (DP)

ANNEXES

Annexe 1-Activité-1- Exemple 1



Annexe 2-Activité 1- Exemple 2

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>La Plateforme_ Network</title>
  <meta name="description" content="Reseau Social pour La Plateforme_" />
  <meta name="keywords" content="stories, posts, social network, followers" />
  <meta name="author" content="Denis Farkas Thuc-nhi Wiedenhofer" />
  <link rel="stylesheet" href="assets/css/style.css">
  <link rel="stylesheet" href="assets/css/bootstrap.css">
  <script src="assets/js/jquery.min.js"></script>
</head>
<body class="bg-light min-vh-100">
  <main class="container mt-5">
    <div class="row d-flex align-items-center">
      <div class="col-xl-2"></div>
      <div class="col-xl-4 col-sm-12">
        
      </div>
      <div class="col-xl-6 col-sm-12">
        
        <form id="formSignIn">
          (...)
        </fieldset>
      </form>
    </div>
  </main>
  <footer id="footer">
    <div class="col-lg-12"> </div>
  </footer>
  <script src="assets/js/jquery.min.js"></script>
  <script src="assets/js/login.js"></script>
  <script src="assets/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

Annexe 3 Activité 1 exemple 3

```
var formL = document.getElementById("formSignIn");

document.addEventListener("DOMContentLoaded", function() {
    formL.addEventListener('submit', e => {
        e.preventDefault();
        login();
    });
});

var serializeForm = function (form) {
    var obj = {};
    var formData = new FormData(form);
    for (var key of formData.keys()) {
        obj[key] = formData.get(key);
    }
    return obj;
};

function login(){
    var login_form = formL;
    var form_data=JSON.stringify(serializeForm(login_form));
    var xhr = new XMLHttpRequest();
    xhr.withCredentials = true;

    xhr.addEventListener("readystatechange", function() {
        if(this.readyState === 4 && this.status == 200) {
            window.location="home.php";
        }else{
            setTimeout(function(){$("#resultat").html("<p>Erreur d'email ou de password.</p>")}, 1000);
        }
    });

    xhr.open("POST", "api/controllers/login.php");
    xhr.setRequestHeader("Content-Type", "text/plain");

    xhr.send(form_data);
}
```

Annexe 4 Activité 2 exemple 1

« Fichier sql Boutique »

```
CREATE TABLE _User(
  Id_User int(11) NOT NULL AUTO_INCREMENT,
  prenom VARCHAR(50),
  nom VARCHAR(50),
  civilite VARCHAR(10),
  telephone VARCHAR(50),
  email VARCHAR(150),
  password VARCHAR(250),
  is_admin int(11) NOT NULL ,
  date_registre DATETIME,
  PRIMARY KEY(Id_User)
);

CREATE TABLE couleur(
  Id_couleur int(11) NOT NULL AUTO_INCREMENT,
  nom_couleur VARCHAR(50),
  image_couleur VARCHAR(50),
  PRIMARY KEY(Id_couleur)
);

CREATE TABLE taille(
  Id_taille int(11) NOT NULL AUTO_INCREMENT,
  nom_taille VARCHAR(50),
  cm_taille VARCHAR(50),
  PRIMARY KEY(Id_taille)
);

CREATE TABLE detail_article(
  Id_detail int(11) NOT NULL AUTO_INCREMENT,
  description TEXT,
  img_palme VARCHAR(50),
  calibre DECIMAL(6,2),
  PRIMARY KEY(Id_detail)
);

CREATE TABLE paiement(
  Id_paiement int(11) NOT NULL AUTO_INCREMENT,
  logo VARCHAR(50),
  nom_paiement VARCHAR(50),
  mode_paiement VARCHAR(50),
  PRIMARY KEY(Id_paiement)
);

CREATE TABLE livraison(
  Id_livraison int(11) NOT NULL AUTO_INCREMENT,
  nom_livreur VARCHAR(50),
  prix_livreur DECIMAL(8,2),
  PRIMARY KEY(Id_livraison)
);


CREATE TABLE commande(
  Id_commande int(11) NOT NULL AUTO_INCREMENT,
  date_commande DATETIME,
  statut_commande int(11) NOT NULL ,
  Id_User INT,
  PRIMARY KEY(Id_commande),
  FOREIGN KEY(Id_User) REFERENCES _User(Id_User)
);
```

DOSSIER PROFESSIONNEL (DP)

Annexe 5 Activité 2 exemple 3

Crud articles administrateur

[BOUTIQUE](#) [Contacts](#) [A notre sujet](#) adm, vous êtes connecté(e) comme Administrateur. [Gestion](#) [Déconnexion](#)



MONTECRISTI

FUNCTIONS

CLIENTS

Liste clients

PRODUITS

Liste produits

Ajout produit

Ajout image produit

ARTICLES

Liste articles

Ajout article

FACTURES

Liste factures

ID	GENRE	NOM	TAILLE	COULEUR	IMAGE	DATE	PRIX	REMISE	QUANTITÉ	ACTION
1	Masculin	Fin Naturel	S	Naturel	mh-exf.jpg	2021-02-12	199	10	19	Modifier
5	Masculin	Extra Fin Naturel	S	Naturel	mh-sf.jpg	2021-02-12	399		20	Modifier
2	Masculin	Fin Naturel	M	Naturel	mh-exf.jpg	2021-02-12	199		20	Modifier
6	Masculin	Extra Fin Naturel	M	Naturel	mh-sf.jpg	2021-02-12	399		20	Modifier
3	Masculin	Fin Naturel	L	Naturel	mh-exf.jpg	2021-02-12	199		20	Modifier
7	Masculin	Extra Fin Naturel	L	Naturel	mh-sf.jpg	2021-02-12	399		20	Modifier
4	Masculin	Fin Naturel	XL	Naturel	mh-exf.jpg	2021-02-12	199		20	Modifier
8	Masculin	Extra Fin Naturel	XL	Naturel	mh-sf.jpg	2021-02-12	399		20	Modifier