



DOSSIER PROFESSIONNEL (DP)

Nom de naissance ▶ Savickaite
Nom d'usage ▶ Savickaite
Prénom ▶ Laura
Adresse ▶ 24 rue gabriel nuvolone, 13013 Marseille

Titre professionnel visé

Concepteur/Développeur d'application 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.

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

Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

p. 5

► Boutique en ligne	p.	p.	5
► Boutique en ligne	p.	p.	9
► Intitulé de l'exemple n° 3	p	p.	

Concevoir et développer la persistance des données en intégrant les recommandations de sécurité

p.

► Chuu Chat	p.	p.	20
► Chuu Chat	p.	p.	23
► Intitulé de l'exemple n° 3	p	p.	

Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

p.

► Chuu Chat	p.	p.	23
► Projet entreprise	p.	p.	31
► Snake	p	p.	34

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

p.

Déclaration sur l'honneur

p. 40

Documents illustrant la pratique professionnelle *(facultatif)*

p.

Annexes *(Si le RC le prévoit)*

p.

EXEMPLES DE PRATIQUE PROFESSIONNELLE

Activité-type 1

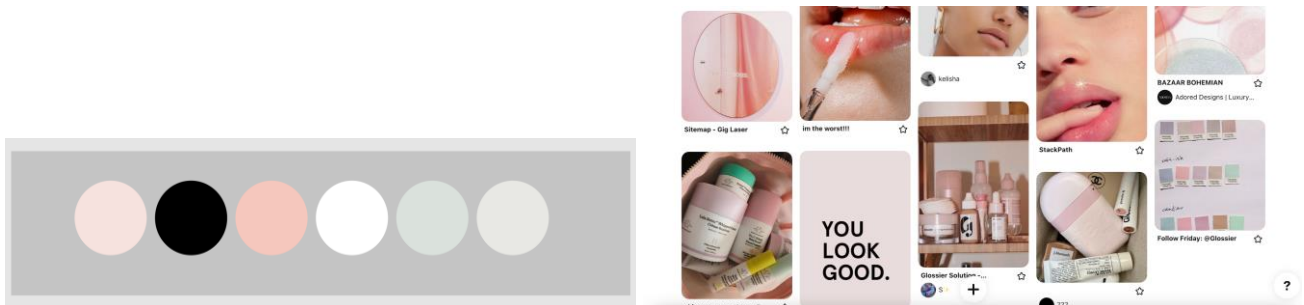
Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

Exemple n°1 ► Boutique en ligne de vente de cosmétiques (Everglow)

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

Maquetter une application

Le projet de Boutique en ligne, réalisé dans le cadre de ma formation à La Plateforme en première année, avait pour but de créer un site e-commerce dans son entièreté (front-end tout comme back-end). Étant un projet de groupe, l'organisation était un point clef à ne pas négliger. Afin d'avoir une harmonie visuelle commune pour chaque page, nous avons réalisé tout d'abord une **charte graphique**. Elle comportait les couleurs principales que nous voulions utiliser dans le site mais aussi un *moodboard* (assemblage visuel).

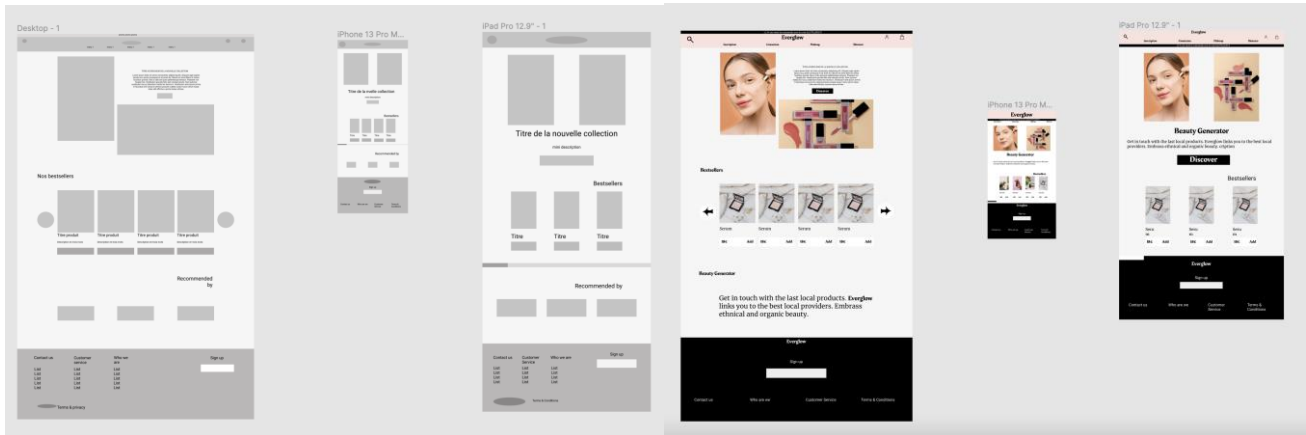


Après une vision globale, nous sommes passés à la partie du **maquettage**. Pour ceci, nous avons utilisé l'outil *Figma*. Nous avons commencé par la **maquette low-fidelity** afin de voir la disposition des éléments dans notre site web mais également, pour gagner du temps. En effet, la maquette low-fidelity (*Lo-Fi*) demande *moins de ressources* (il peut être fait avec un crayon et un papier même) et permet une communication *plus rapide* avec le client (si le design est rejeté, on peut refaire une maquette Lo-Fi rapidement sans passer énormément de temps et d'énergie au cas où celle-ci soit encore rejetée).

Une fois cette dernière validée par le client ou bien les membres du projet, on peut passer à la **maquette high-fidelity** (*Hi-fi*) dont le design final du site se rapprochera le plus esthétiquement

DOSSIER PROFESSIONNEL (DP)

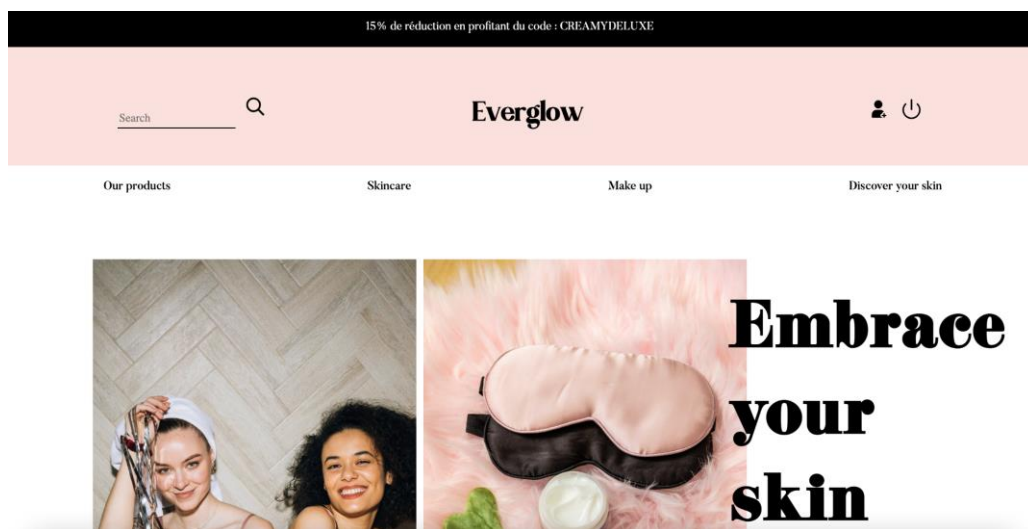
mais également fonctionnellement.



Maquette lo-fi de la page d'accueil sous trois formats (desktop, mobile, tablette) Maquette hi-fi de la page d'accueil sous trois formats (desktop, mobile, tablette)

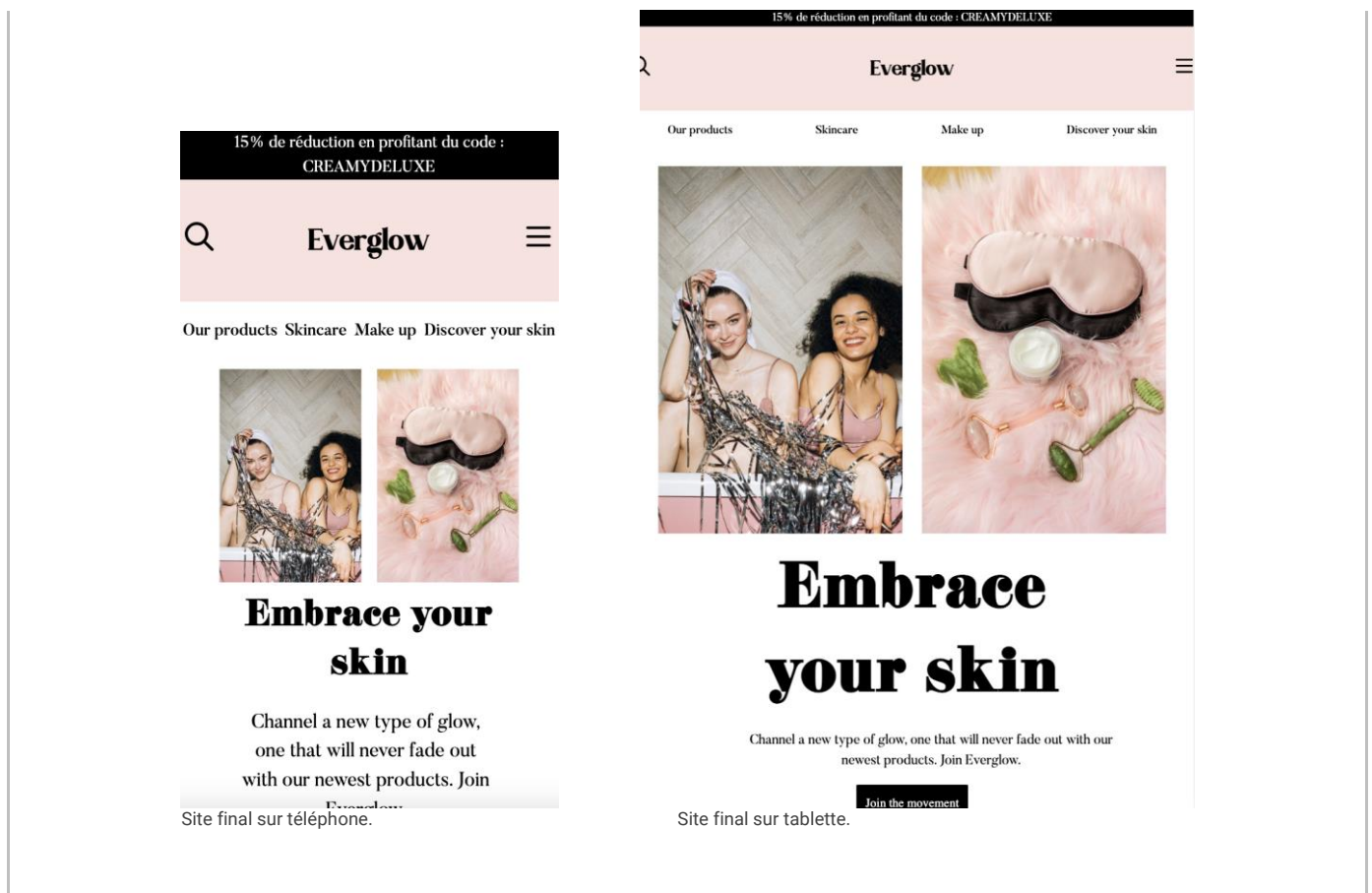
Par la suite, nous avons codé l'interface en utilisant du **Javascript** pour le *responsive* de la barre de navigation par exemple : si l'utilisateur est sur un téléphone, il aura un *menu burger* afin de naviguer parmi les différentes pages du site tandis que s'il est sur ordinateur, ce menu burger laissera place à un *menu plus classique* étalé sur la longueur de la nav barre.

Nous avons également utilisé de l'**HTML** et du **CSS** et plusieurs tests ont été réalisés afin de voir si les éléments étaient bien disposés.



Site final sous desktop.

DOSSIER PROFESSIONNEL (DP)



2. Précisez les moyens utilisés :

- Figma pour la maquette
- Visual Studio Code pour l'édition de code
- Github
- HTML/CSS/JS pour l'intégration
- Media Queries pour le responsive

3. Avec qui avez-vous travaillé ?

Max Machin, [Laura Scognamiglio](#)

4. Contexte

Nom de l'entreprise, organisme ou association ► La Plateforme

Chantier, atelier, service ► Boutique en ligne.

Période d'exercice ► Du : 2022-02-21 au : 2022-03-07

DOSSIER PROFESSIONNEL ^(DP)

5. Informations complémentaires *(facultatif)*

Activité-type 2

Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

Exemple n° 1 ► Boutique en ligne (Everglow)

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

Développer la partie front-end d'une interface utilisateur web;

Développer la partie back-end d'une interface utilisateur web;

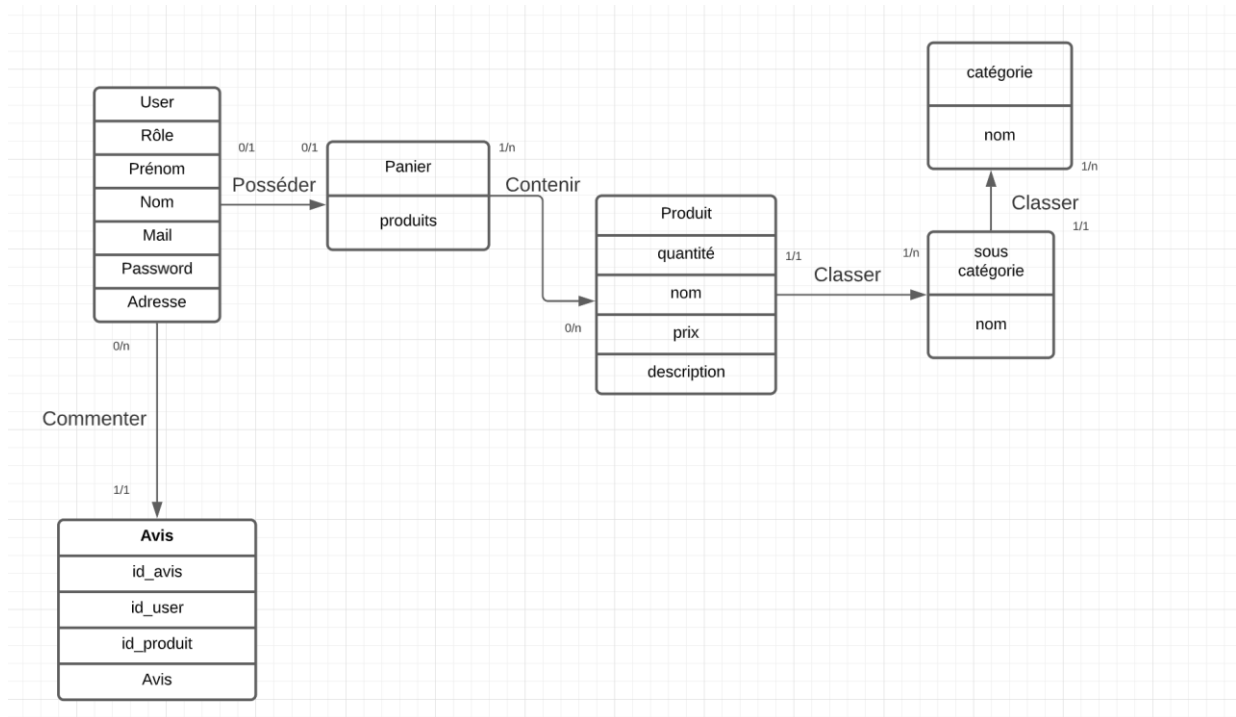
Concevoir une base de données;

Mettre en place une base de données;

Le projet de Boutique en ligne, réalisé dans le cadre de ma formation à La Plateforme, avait pour but de créer un site e-commerce dans son entièreté (front-end tout comme back-end) : dont le **CRUD** (create, read, update, delete). Étant un projet de groupe, l'organisation était un point clef à ne pas négliger.

Pour se faire, nous avons utilisé la méthode *MERISE* afin d'arriver à une conception d'un système d'information. Parmi cette méthode, on a employé le *MCD* (modèle conceptuel des données) et *MLD* (modèle logique des données), le *MPD* (modèle physique des données) n'ayant pas été établi jusqu'à la fin.

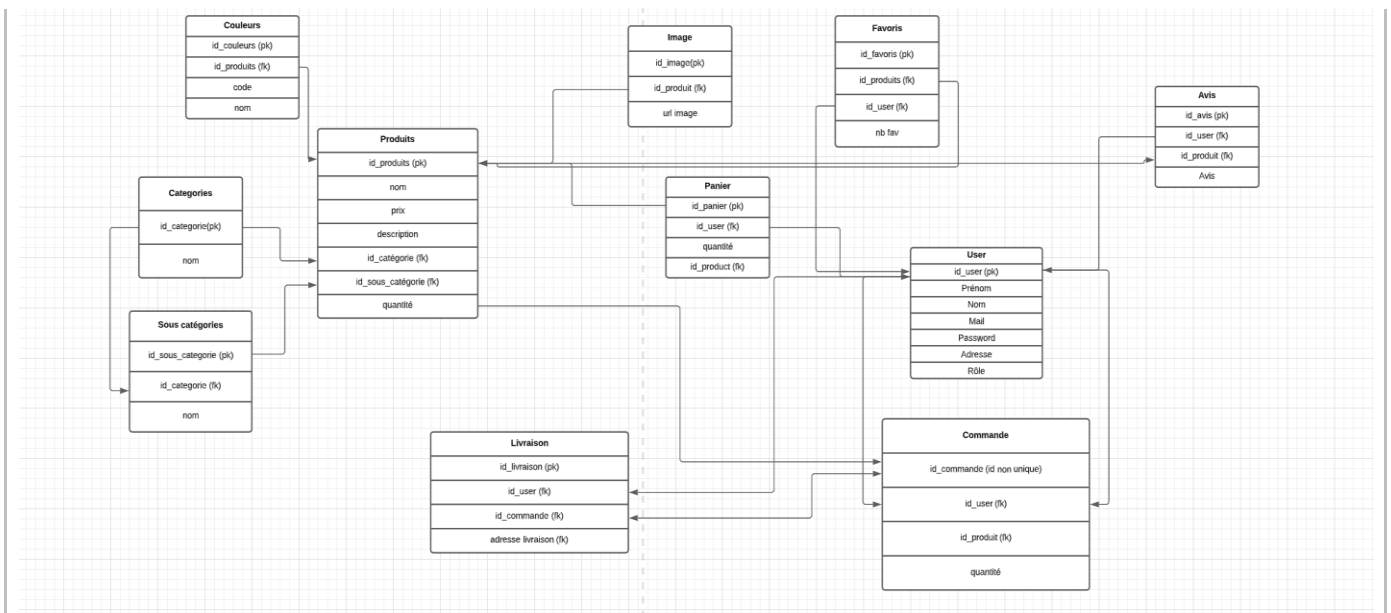
DOSSIER PROFESSIONNEL (DP)



Modèle MCD de notre boutique en ligne.

Le MCD comporte un début de base de données. Il se compose des informations principales (pas de tables de liaison), de leurs relations entre elles (*cardinalités* traduisant les *règles de gestion**) et de verbes à l'infinitif traduisant ces liens. Le MCD permet une visualisation précoce de la future base de données. Pour préciser cette vision, il est utile de réaliser un MLD sur cette première base.

DOSSIER PROFESSIONNEL (DP)



Modèle MLD de notre boutique en ligne.

Le MLD devrait comporter la plupart des tables : il comprend celles de liaison et les *foreign keys* (une clé faisant référence à la primary key d'une autre table) qui les relient entre elles ainsi que les *primary keys*. Le MLD est voué à évoluer au fur et à mesure du projet mais généralement il représente l'organisation des tables - c'est également à ce stade qu'il est possible de savoir quelles tables seront utiles et devront être créées, combien sont-elles etc...

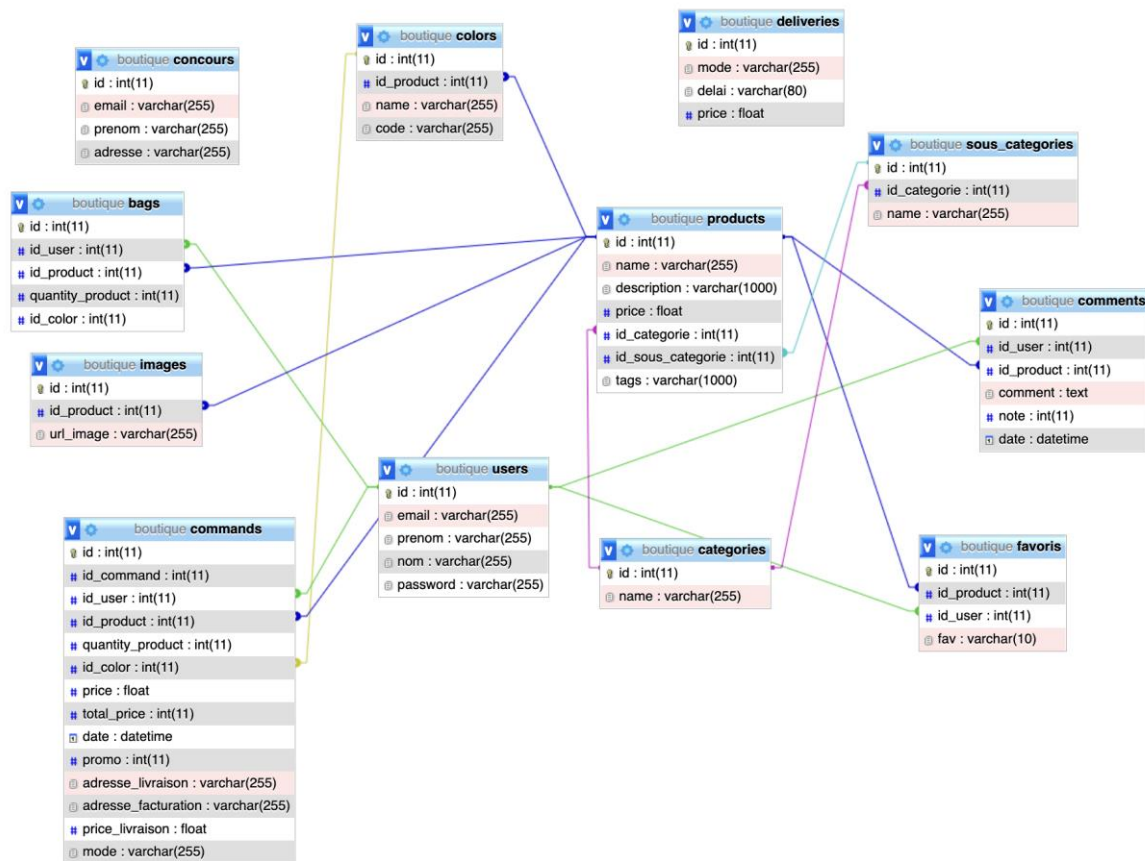
Nous avons donc :

- la table User qui recueillait toutes les données des utilisateurs;
- la table Produits qui recueillait toutes les informations des produits;
- les tables Panier et Commande relient les produits aux users (une fois validé, le panier devient une commande);
- la table Livraison relie les users aux commandes correspondantes;
- les tables Couleurs et Image sont liées aux produits et leur donnent une spécificité, si Image est obligatoire, Couleurs dépendra du produit;
- les tables Catégorie et Sous catégories rassemblent les produits et les organisent;
- les tables Favoris et Avis relient les produits et les users et permettent aux

DOSSIER PROFESSIONNEL (DP)

users de communiquer à propos des produits.

Aussi cette méthode nous a permis de créer une base de données fonctionnelle pour l'ensemble du projet ayant évolué tandis qu'on continuait le reste des opérations.



Visuel de la database de notre Boutique en Ligne.

Pour accéder à l'ensemble du site, il faut s'inscrire et/ou se connecter.

Ensuite, si l'utilisateur est connecté, il a accès à son profil qui rassemble toutes les commandes qu'il avait passé, ses favoris ainsi qu'à ses commentaires. Il peut, évidemment, visiter le site entièrement et sélectionner des produits à mettre dans son panier. Pour simuler un paiement nous avons utilisé *STRIPE* (API de paiement). Certains produits ont différentes couleurs que l'utilisateur peut sélectionner, il peut les rajouter dans ses favoris et laisser un commentaire

DOSSIER PROFESSIONNEL ^(DP)

(avec système de notation en étoiles).

Lorsque l'Admin est connecté, il a accès à son dashboard duquel il peut voir les produits les plus populaires, il peut en ajouter, en supprimer, en modifier.

- L'utilisateur ajoute un objet dans son panier : le bouton a un input hidden qui détient l'id du produit, ce dernier ainsi que celui de l'utilisateur vont être insérés dans la table Panier. Pour le supprimer, c'est la même chose, le bouton a un input hidden de l'id produit et la ligne est supprimée de la table.
- L'utilisateur veut voir toutes ses commandes : elles lui sont accessibles via son id.
- L'admin veut rajouter un produit : il remplit un formulaire qui sera inséré dans la base de données. Pour ajouter des images, il sélectionne le produit voulu et intègre des images qui doivent correspondre aux formats acceptés (jpeg, jpg, png).
- L'admin veut modifier un produit : il a deux choix, il peut soit aller directement sur la page du produit et le modifier -> ça le ramène à la page formulaire pré-remplie avec les données déjà existantes (grâce à son id), soit il va dans la liste des produits et sélectionne celui qu'il veut modifier (idem pour le formulaire).

L'ensemble du projet a été réalisé en **objets** qui correspondaient plus ou moins aux tables que nous avions.

Il y avait :

- la classe User
- la classe Products
- la classe Bags
- la classe Commands
- les classes Catégories et Sous-Catégories
- la classe Images et celle Couleurs
- la classe Admin
- la classe Favoris

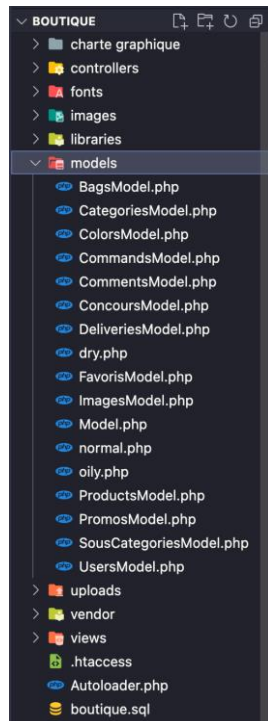
DOSSIER PROFESSIONNEL (DP)

- la classe Deliveries

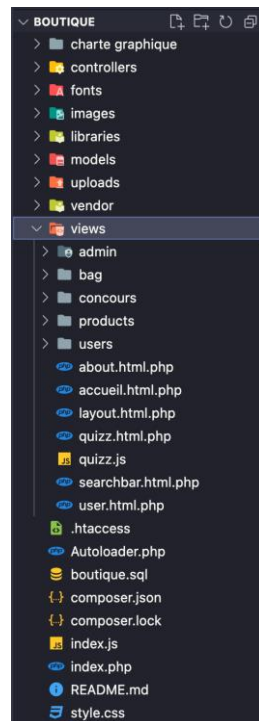
Pour organiser tout ceci, le *design pattern* du **MVC** (Model View Controller) a été mis en place. Les *Models* (server side) servaient à prendre les informations de la base de données, tandis que les *Controllers* manipulaient ces informations. Les *Views* les présentaient à l'utilisateur (user side). Afin de réaliser un réel MVD, j'ai mis en place un **Routeur** et son **Autoloader**.

Le routeur récupérait l'URL, l'explorait pour récupérer un élément qui lui permettait de récupérer le Controller demandé. Le controller, lui, s'occupait de prendre le model qui lui correspondait. On devait préciser à chaque route sa view.

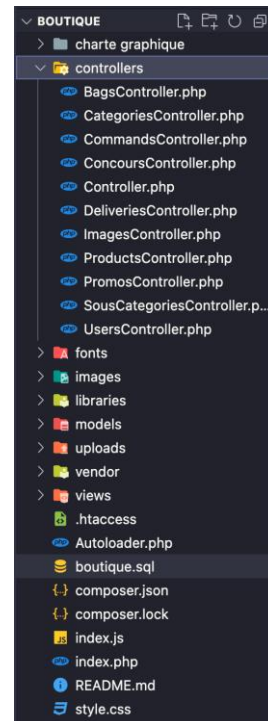
L'autoloader, lui, permettait le chargement dynamique des classes appelées lors des actions. Il y avait également un **Renderer** qui s'occupait de projeter la view sous forme de "tampon" avec *ob_start* et *ob_clean* entre un *layout* qui était composé d'un *header* et *footer*.



Les Models.



Les Views.



Les Controllers.

J'ai aussi intégré l'**API Facebook** (la partie Instagram) au site en reliant à un compte Instagram fictif. Pour ceci je me suis servi de **Postman** pour accéder à mes *requêtes GET*. Il me fallait un

DOSSIER PROFESSIONNEL (DP)

token unique récupéré auprès du compte Instagram voulu. Une fois que je l'avais, il me suffisait de copier/coller les requêtes trouvées dans les documentations sur Postman, et de remplacer les clés par mes valeurs.

Exemple : GET

`https://graph.instagram.com/me/media?fields={fields}&access_token={access-token}`

Les **fields** étaient ce que je voulais récupérer, par exemple `media_type` et `media_url`. L'**access token** était mon token unique récupéré précédemment.

```
const accessUrl = `https://graph.instagram.com/me/media?fields=${fields}&access_token=${accessToken}`;

const section = document.getElementsByClassName("instafeed");
const instafeed = section[0];

const fetchPosts = async () => {

  try {

    const response = await fetch(accessUrl);
    const {data} = await response.json();

    data.forEach(post => {
      let a = document.createElement("a");
      a.href = post.permalink;
      a.target = "_blank";
      a.rel = "noreferrer noopener";

      if(post.media_type === "VIDEO"){
        let video = document.createElement("video");
        video.src = post.media_url;
        video.preload = true;
        video.autoplay = true;
        video.muted = true;
        video.loop = true;
        a.appendChild(video);
      } else {
        let img = document.createElement("img");
        img.src = post.media_url;
        a.appendChild(img);
      }

      instafeed.appendChild(a);
    });
  }
};
```

Morceau de code de la manipulation API instagram.

J'attendais un **JSON** de ma requête **GET**. Une fois le **JSON** trouvé, je pouvais le manipuler : retirer l'*url de l'image du Post Instagram* et l'afficher dans mes balises `img` (dans le `src`). Si jamais c'était une vidéo (que l'on voit par rapport au `media_type` récupéré au-dessus) je faisais la même chose sauf que je devais la changer d'abord en la mutant par exemple. Les liens (`a`) devaient être sécurisés avec le "noreferrer noopener" (noreferrer est pour Mozilla Firefox) en `target_blank` (afin d'ouvrir le lien dans une nouvelle page).

Pour l'interface front, nous avons utilisé **HTML**, **CSS** et **Javascript** vanilla.

JE ME CONNECTE

E-mail *

b@a.com

Mot de passe *

.

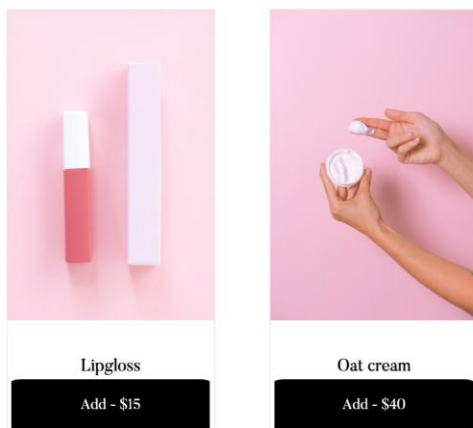
Mot de passe oublié ?

CONNEXION

Pas encore inscrit ? [Rejoignez-nous](#)

L'utilisateur pouvait se connecter à son compte et retrouver son profil avec l'ensemble des produits qu'il avait en "favoris" et les acheter directement.

Vos produits favoris



Il pouvait également trouver d'autres produits à acheter en utilisant la barre de navigation. Aussi, s'il voulait acheter du make-up - il lui suffisait de cliquer sur "make-up".

DOSSIER PROFESSIONNEL ^(DP)

Our products

Skincare

Make up

Discover your skin

lips

eyes

face



Pour plus de précision dans la recherche, l'utilisateur pouvait trouver des sous-catégories.



LIPGLOSS 

Moisturizing lipgloss for your lips. No more chapped lips in the winter. Get glossy!



15 €

1

Add

★★★★★

ok

Posté par : aze le : 01/04/2022 à 14:08:57

L'icone coeur du lipgloss est coloré, ce qui signifie que le produit était déjà dans les favoris - dans le cas contraire, le coeur aurait été blanc.

Le choix de la quantité et de la couleur du lipgloss (ainsi que de la mise en favoris) était géré en Javascript.

DOSSIER PROFESSIONNEL ^(DP)

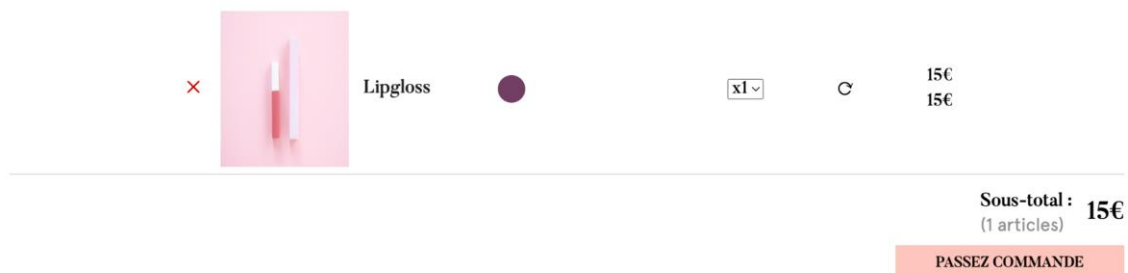


Une fois qu'on avait cliqué sur "add", une notification apparaissait (en javascript encore une fois) pour prévenir que le produit était dans le panier.

👍 Produit ajouté au panier

On pouvait également le supprimer du panier ou changer le nombre souhaité.

Mon panier



***règles de gestion** : règles régissant l'association de deux objets, elles sont préétablies par l'entreprise généralement. (ex : un client est-il considéré comme tel lorsqu'il a un panier ou peut-il être un client sans ? sous-entendu, prenons-nous les prospects ?)

2. Précisez les moyens utilisés :

- **Figma** pour la maquette
- **Lucidchard** pour les MCD/MLD
- **Visual Studio Code** pour l'édition de code
- **Github**
- **HTML/CSS** pour l'intégration
- **Media Queries** pour le responsive
- **Javascript** pour l'interactivité user-side + mon fetch d'API
- **Postman** pour les requêtes API
- **API Instagram**
- **JSON**
- **Stripe**

DOSSIER PROFESSIONNEL (DP)

3. Avec qui avez-vous travaillé ?

Max Machin, [Laura Scognamiglio](#)

4. Contexte

Nom de l'entreprise, organisme ou association ► *La Plateforme_*

Chantier, atelier, service ► *Boutique en ligne.*

Période d'exercice ► Du : 2022-02-21 au : 2022-03-07

5. Informations complémentaires (facultatif)

Activité-type 3

Concevoir et développer la persistance des données en intégrant les recommandations de sécurité
Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

Exemple n° 1 ► Chuu Chat (application chat mobile)

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

Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement **Concevoir une application**

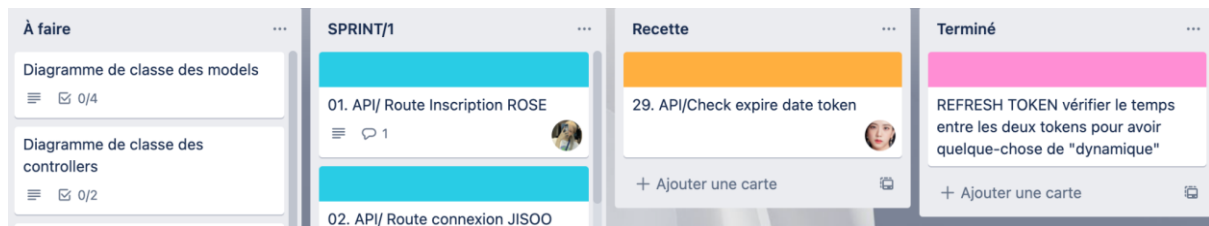
Chuu chat était une application mobile de messagerie instantanée, réalisée dans le cadre de ma seconde année au sein de l'école. Le but était de concevoir et de développer une application mobile en groupe.

Pour ceci, l'organisation était primordiale - le projet ayant été lancé quelques mois après le début de notre alternance, nous avons pu mettre en place des bonnes pratiques rencontrées au sein de nos entreprises respectives.

Etant donné que notre temps était saccadé entre le temps en entreprise et celui à l'école, nous avons besoin de flexibilité ce que la méthode **AGILE/SCRUM** nous a permis d'avoir. C'est un cadre de gestion de projet qui divise les projets en plusieurs phases dynamiques appelées "sprints". Après chaque **sprint**, on parlait des axes d'améliorations, des éléments bloquants et des éléments ralentisseurs ("parachute") à prendre en compte pour le sprint suivant. Nous avons également des "**daily**" - des petites réunions qui nous permettaient de faire un point chacun sur les tâches accomplies et celles qu'on comptait réaliser le jour-même - aussi, si nous rencontrions un problème, on n'avait pas à attendre la fin du sprint pour en faire part. Cela construisait également une compréhension commune du produit avec un partage d'informations.

Les sprints étaient également l'occasion de revoir les **tickets**/d'en créer de nouveau. Les tickets étaient des demandes à réaliser. Cela découpait le projet en plusieurs petites parties. On utilisait Trello pour les manager.

DOSSIER PROFESSIONNEL (DP)



Capture d'écran de notre trello (cycle de vie d'un ticket).

Le ticket était décomposé en trois parties : la demande initiale (celle du client), la description et les “besoins” ou solutions potentielles pour réaliser le ticket.



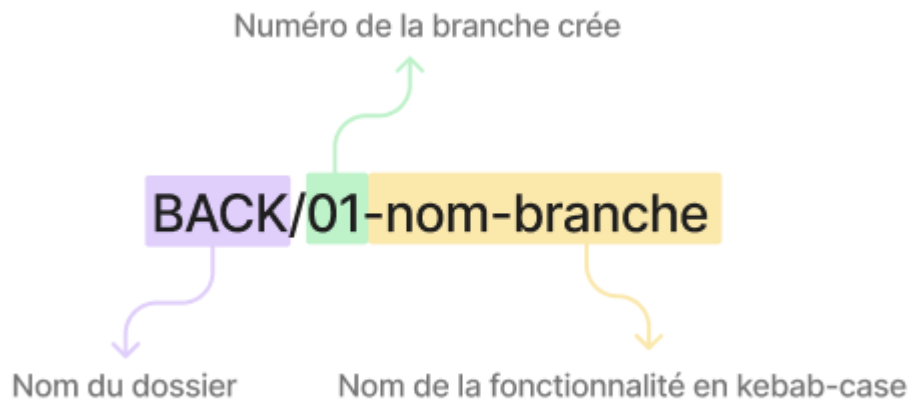
Exemple d'un ticket.

Il y avait également une organisation du git et de la nomenclature des branches. Nous avions une branche par ticket. Une fois une fonctionnalité complètement développée, on “mergeait” la branche correspondante vers une branche de “pré-production” - il y a eu plusieurs branches “pré-production” qui concordent avec le numéro du sprint en cours. Enfin nous avions une branche dite de production (la branche “main”) qui recevait principalement les merges considérés comme “finaux” - en d’autres termes, lorsque les différents merges sur un sprint fonctionnaient ensemble, on “mergeait” sur main.

La nomenclature, elle, permettait de nous retrouver au niveau des tickets : ils portaient donc une

DOSSIER PROFESSIONNEL ^(DP)

indication à ces derniers.



2. Précisez les moyens utilisés :

- **Trello** pour l'organisation des tickets
- **Git**

3. Avec qui avez-vous travaillé ?

Naomi Monderer, [Laura Scognamiglio](#), [Dorian Palace](#)

4. Contexte

Nom de l'entreprise, organisme ou association ► *La Plateforme_*

Chantier, atelier, service ► *Application mobile chat.*

Période d'exercice ► Du : 2023-01-05 au : 2023-06-30

5. Informations complémentaires (facultatif)

Activité-type 4

Concevoir et développer la persistance des données en intégrant les recommandations de sécurité
Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

Exemple n° 1 ► Chuu Chat (application chat mobile)

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

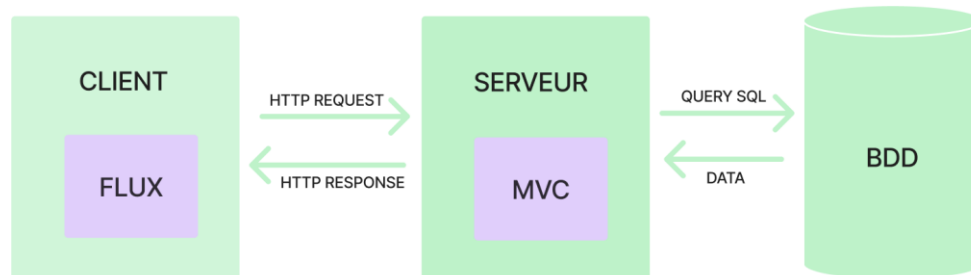
**Construire une application organisée en couches;
Développer une application mobile;
Développer des composants dans le langage d'une base de données;
Développer des composants métier**

Chuu chat était une application mobile de messagerie instantanée, réalisée dans le cadre de ma seconde année au sein de l'école. Le but était de concevoir et de développer une application mobile en groupe.

Pour construire notre application, il a fallu choisir une architecture. Le modèle initial est celui de **client-serveur** : deux couches distinctes qui communiquent ensemble. Le serveur fournit des ressources et des services à plusieurs clients, qui peuvent être situés sur des ordinateurs différents. Les clients communiquent avec le serveur en utilisant un protocole standard, tel que celui HTTP. Nous avons donc finalement travaillé une **architecture trois-tiers** où il y a une réelle séparation de couches, entre non seulement le client et l'application mais également la base de données.

DOSSIER PROFESSIONNEL ^(DP)

Architecture 3-Tiers

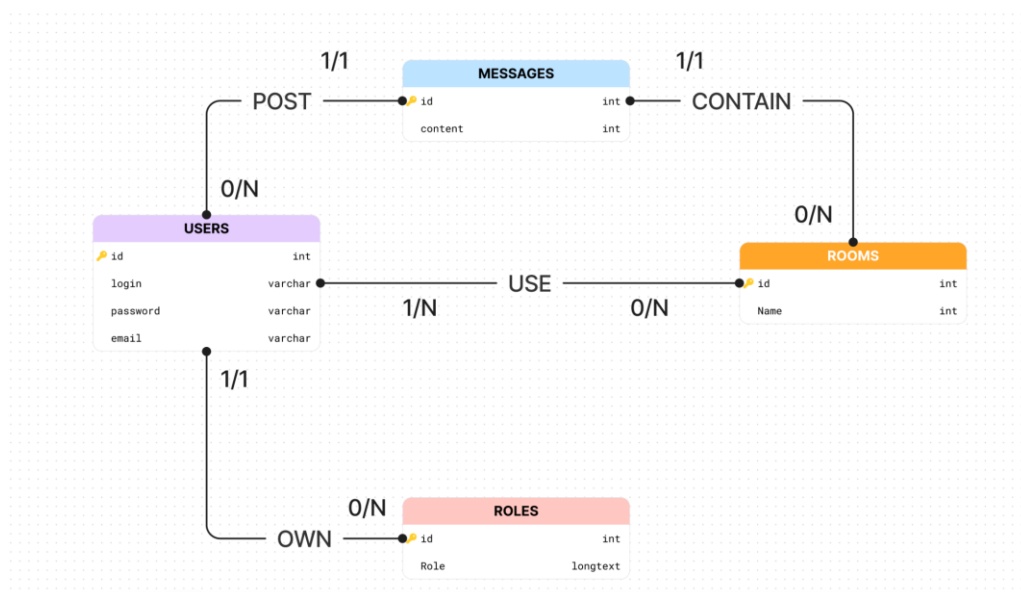


La séparation des composants d'application en couches distinctes renforce la facilité de maintenance et l'évolutivité de l'application.

Notre **API** était donc de type **REST** qui permettait à différents systèmes de communiquer entre eux en utilisant des méthodes HTTP standardisées comme *GET* (récupérer une donnée), *POST* (enregistrer une donnée), *PUT* (mettre à jour l'intégralité des informations d'une donnée), *PATCH* (mettre à jour partiellement une donnée) et *DELETE* (supprimer une donnée).

Une fois cela convenu, nous avons conceptualisé la base de données.

Premièrement avec un MCD :

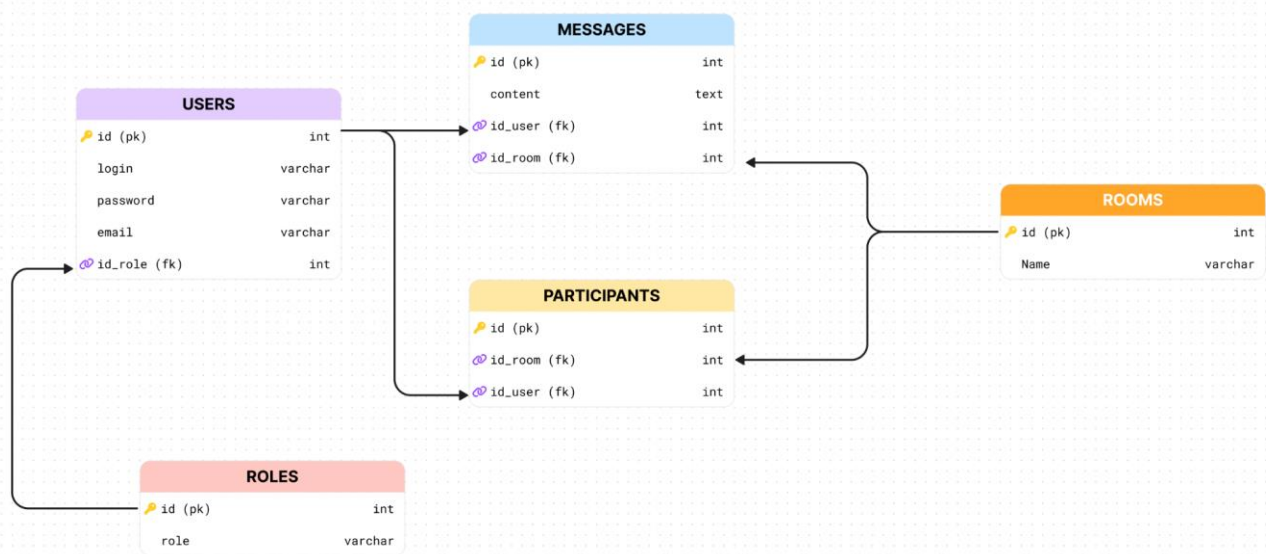


MCD.

DOSSIER PROFESSIONNEL ^(DP)

Dans le modèle ci-dessus, un utilisateur peut utiliser 0 à plusieurs chatrooms tandis qu'une chatroom peut exister sans utilisateurs comme avec plusieurs. Un utilisateur n'est pas obligé de poster un message comme il peut en poster plusieurs tandis qu'un message est obligatoirement et uniquement posté par un utilisateur etc..

Par la suite, sur cette base-ci, nous avons réalisé le MLD :

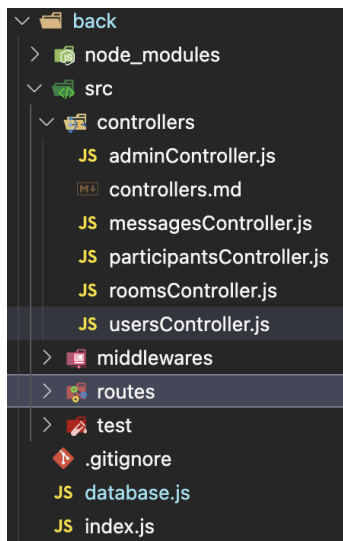


MLD

Dans le modèle ci-dessus, nous avons mis en place une table de liaison entre les utilisateurs et les chatrooms sous le nom de participants (la relation étant de nature N/N) avec les id des rooms mais également ceux des utilisateurs.

Pour organiser l'application, nous avons choisi de modifier la structure classique du **MVC**. En effet, alors que nous avons bien un controller, nous avons trouvé cela plus facile de ne pas séparer le model du controller et donc avoir un seul dossier. La view est, quant à elle, gérée par le front.

DOSSIER PROFESSIONNEL (DP)



Architecture de notre back

Aussi notre partie API comportait les controllers et les routes qui menaient à ces derniers et servaient aux différentes requêtes distribuées par le côté client.

```
//[BACK/03-get-all-users]: Une route qui  
router.get('/', getUsers);
```

Morceau de code d'un appel à une route.

```
const getAllFromUsers = (req, res) => {  
  const sql = 'SELECT * FROM users'  
  db.query(sql, function (error, data) {  
    if (error) throw error;  
    else res.send(data);  
  })  
}
```

Morceau de code de la fonction..

Pour se connecter à la base de données, nous avons installé le module sql dans notre projet et l'avons instantié. Pour se connecter à la base de données nous avons utilisé la méthode `createConnection()` permettant de passer les paramètres `host`, `password`, `database` et `port` afin de rentrer les valeurs correspondantes et d'établir la connexion à la base de données.

```
const mysql = require('mysql');
var connection
connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: 'blackpinkinyourarea4',
  database: 'chat',
  port: '8889',
});

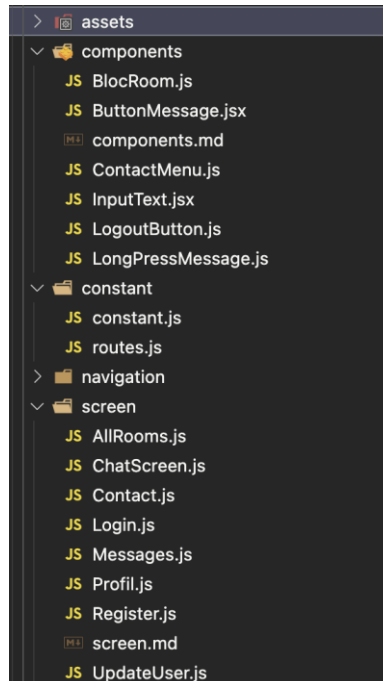
connection.connect(function(err) {
  if (err) {
    console.error('Error connecting to database:', err);
    return;
  }
  console.log(`Connected to database on port ${connection.config.port}!`);
});

module.exports = connection
```

Morceau de code de la connexion à la bdd.

Le front, quant à lui, était également divisé en plusieurs dossiers :

DOSSIER PROFESSIONNEL (DP)



Architecture du front..

Ayant choisi de réaliser le front en React Native, il était question de développer avec des composants qu'on pouvait réutiliser. Les screens utilisaient les dits composants. Il existe des composants en class mais aussi en **fonctions** - dans le cadre de notre application, nous avons utilisé les derniers, étant recommandés puisque préférés dernièrement dans le développement.

```
export default function BlocRoom({name, room, id, pressRoom, deletePress, specialClass, touchable, tb, disabled, n

  const blocRoomPress = () => {
    // console.log('room id', room.id)
    pressRoom({'id':room.id, 'name':name})
  }

  const deleteRoom = () => {
    deletePress(id)
  }

  return (
    //ajouter dans le press qu'on peut aller dans les messages de la room également si c'est notre room
    <TouchableOpacity
      style={styles.container}
      onPress={() => {deletePress ? deleteRoom() : blocRoomPress()}}
      // disabled={}
    >
      { /* Image */ }
      <View style={styles.tinyIcon}>
        <Image
          style={styles.img}
          source={{uri: 'https://64.media.tumblr.com/6b9e50e7237cf04fd44b6f56ce75e848/04f19f3b5d21afac-b
        />
      </View>
      <Text style={styles.name}>{name}</Text>
    </TouchableOpacity>
  )
}
```

DOSSIER PROFESSIONNEL (DP)

Morceau de code d'un composant.

Ce composant était par ailleurs réutilisé dans une view complète.

```
<ScrollView style={styles.bg}>
  <View style={styles.tabs}>
    <TouchableOpacity onPress={() => underlined(2)}>
      <Text style={underline === 2 ? styles.selected : styles.notSelected}>More bands</Text>
    </TouchableOpacity>
    <TouchableOpacity onPress={() => underlined(1)}>
      <Text style={underline === 1 ? styles.selected : styles.notSelected}>My chuu bands</Text>
    </TouchableOpacity>
  </View>
  <View style={styles.container}>
    {
      rooms.length >= 1 ?
        rooms.map((room) =>
          <BlocRoom
            key={room.id}
            name={room.name}
            room={room}
            tab={underline}
            pressRoom={setNewRooms}
            // specialClass={moreRooms.find(moreRoom => moreRoom.id === room.id) ? true : false}
            // touchable={touchable}
            tb = {false}
            disabled = {disabled}
            moreRooms = {moreRooms}
          />
        )
      :
      <Text>No rooms.</Text>
    }
  </View>
```

Morceau de code d'une view avec le composant BlocRoom consommé.

Ce système permettait d'avoir un code plus facile à maintenir et à faire évoluer si on le souhaitait.

2. Précisez les moyens utilisés :

- Trello pour l'organisation des tickets
- Git

3. Avec qui avez-vous travaillé ?

Naomi Monderer, [Laura Scognamiglio](#), [Dorian Palace](#)

DOSSIER PROFESSIONNEL ^(DP)

4. Contexte

Nom de l'entreprise, organisme ou association ► *La Plateforme_*

Chantier, atelier, service ► *Application mobile chat.*

Période d'exercice ► Du : 2023-01-05 au : 2023-06-30

5. Informations complémentaires *(facultatif)*

Activité-type 5

Concevoir et développer la persistance des données en intégrant les recommandations de sécurité

Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

Exemple n° 1 ► *Projet entreprise d'une camarade*

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

Préparer et exécuter les plans de tests d'une application;

Pour cette partie, j'ai décidé de me renseigner auprès d'une camarade qui a pu faire des **tests unitaires** avec son entreprise. Les tests unitaires consistent à tester de manière individuelle et isolée chaque fonction du projet en comparant le résultat effectif et le résultat attendu en fonction d'un jeu de données test prédéfinies.

Dans son cas, elle a utilisé **Jest** (framework de test pour Javascript) pour l'aider dans la formulation et l'exécution de ses tests. Pour cela, il a fallu installer non seulement Jest mais également une librairie qui adaptera le framework au langage testé - dans son cas c'était adapté pour React (*react-script test*).

DOSSIER PROFESSIONNEL (DP)

```
import dayjs from 'dayjs';

export const effectivePeriodSchema = (startDate, endDate, formatMessage) => {
  const currentDate = dayjs().format();

  let errors = {};

  if (startDate) {
    if (dayjs(startDate).isBefore(currentDate)) {
      errors.startDate = formatMessage({ id: 'qedit.section.StartError' });
    } else if (dayjs(startDate).isSame(currentDate)) {
      errors.startDate = formatMessage({
        id: 'qedit.section.StartCurrentError',
      });
    }
  }

  if (endDate) {
    if (dayjs(endDate).isBefore(currentDate)) {
      errors.endDate = formatMessage({ id: 'qedit.section.EndError' });
    } else if (dayjs(endDate).isSame(currentDate)) {
      errors.endDate = formatMessage({ id: 'qedit.section.EndCurrentError' });
    }
  }
}
```

La fonction testée se nommait `effectivePeriodSchema` - elle avait trois paramètres, `startDate`, `endDate` et `formatMessage`. Il fallait donc tester tous les cas possibles pour cette fonction (le `startDate` non renseigné ou le `endDate`, ou les deux renseignés voire le contraire etc...) - toutes les différentes branches de "if".

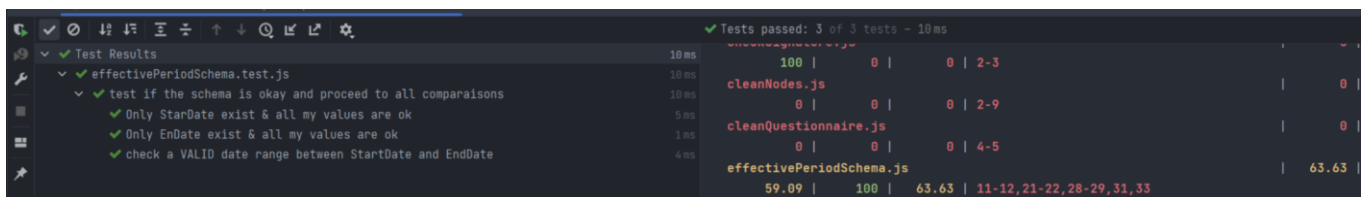
```
effectivePeriodSchema.test.js x effectivePeriodSchema.js x EffectivePeriod.js x
1 import { effectivePeriodSchema } from './effectivePeriodSchema';
2
3 const TEST_DATE = '2023-05-19T00:00:00+02:00';
4 const FORMAT_MESSAGE = { id: "je suis un message d'erreur" };
5
6 describe( name: 'test if the schema is okay and proceed to all comparaisons', fn: () => {
7   test( name: 'Only StarDate exist & all my values are ok', fn: () => {
8     let startDate = TEST_DATE;
9     let endDate = '';
10
11     let errors = effectivePeriodSchema(
12       startDate,
13       endDate,
14       formatMessage: () => FORMAT_MESSAGE,
15     );
16
17     expect(errors).toHaveProperty( propertyPath: 'startDate');
18     expect(startDate).toEqual(startDate);
19     expect(endDate).toBe( expected: '' );
20     expect(FORMAT_MESSAGE).toEqual( expected: { id: "je suis un message d'erreur" } );
21   });
22 }
```

Pour ce faire, elle crée un fichier exprès où elle importe la fonction testée. *Describe* est inhérent à Jest - il n'est là que pour décrire le but de ce test, cette fonction prend en son sein autant de

DOSSIER PROFESSIONNEL (DP)

fonctions “tests” (également inhérentes) que nécessaire. Dans test, on y décrit le test mais également ce qui est attendu comme résultat (avec les *expect* et *.toHaveProperty* etc..). Par la suite, on peut lancer les tests.

Jest nous précisera non seulement les résultats de ces tests mais également leur portée dans le projet ainsi que la couverture de tous les cas possibles.



File	Coverage
cleanNodes.js	0 0 0 2-3
cleanQuestionnaire.js	0 0 0 2-9
effectivePeriodSchema.js	59.09 100 63.63 11-12, 21-22, 28-29, 31, 33

Ici on peut voir qu’il sont tous passés, cependant ils ne répondent pas à tous les cas, seulement à 63.63% (Jest précise les lignes non couvertes, ici 11-12 etc...).

2. Précisez les moyens utilisés :

- Jest
- React
- IntelliJ (ide)
- React-script-test

3. Avec qui avez-vous travaillé ?

[Laura Scognamiglio](#)

4. Contexte

Nom de l’entreprise, organisme ou association ► **ENOVACOM**

Chantier, atelier, service ► //

Période d’exercice ► Du : 2023-07-13 au : 2023-07-14

5. Informations complémentaires (facultatif)

Activité-type 6

Concevoir et développer la persistance des données en intégrant les recommandations de sécurité
Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

Exemple n° 1 ► Snake

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

Développer une interface utilisateur de type desktop;

Dans le cadre de mon année scolaire, nous avons dû créer une application desktop exécutable en Python de type Snake.

La première étape a été d'installer Python.

Download the latest version for macOS

Download Python 3.11.4

Extrait de l'installation de python.

Le jeu de snake tourne sur le terminal car il n'y a pas d'éléments graphiques - dans le cas où je voudrai l'améliorer, je pourrai utiliser PyGame par exemple qui est une bibliothèque facilitant le développement de jeux vidéos en python avec des fonctions déjà intégrées.

```
import random
import curses

s = curses.initscr()
curses.curs_set(0)
sh, sw = s.getmaxyx()
w = curses.newwin(sh, sw, 0, 0)
w.keypad(1)
w.timeout(100)

snk_x = sw//4
snk_y = sh//2
snake = [
    [snk_y, snk_x],
    [snk_y, snk_x-1],
    [snk_y, snk_x-2]
]

food = [sh//2, sw//2]
w.addch(int(food[0]), int(food[1]), curses.ACS_PI)

key = curses.KEY_RIGHT

while True:
    next_key = w.getch()
    key = key if next_key == -1 else next_key

    if snake[0][0] in [0, sh] or \
        snake[0][1] in [0, sw] or \
        snake[0] in snake[1:]:
        curses.endwin()
        quit()

    new_head = [snake[0][0], snake[0][1]]

    if key == curses.KEY_DOWN:
        new_head[0] += 1
    if key == curses.KEY_UP:
        new_head[0] -= 1
    if key == curses.KEY_LEFT:
        new_head[1] -= 1
    if key == curses.KEY_RIGHT:
        new_head[1] += 1

    snake.insert(0, new_head)
```

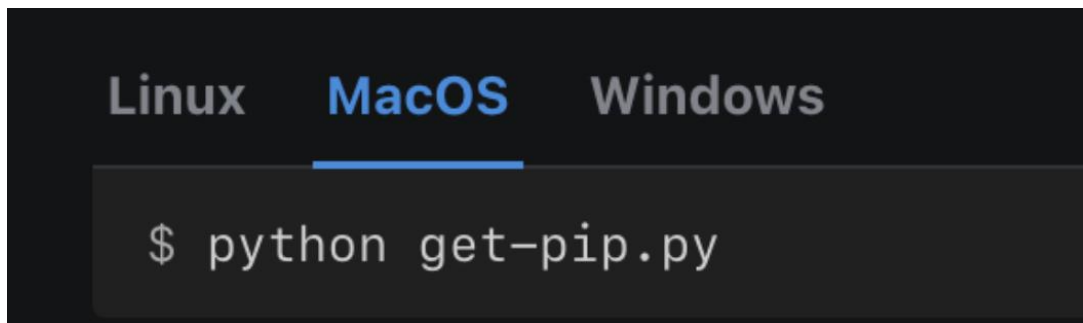
On commence par set la width et la height de notre fenêtre avec les données de sh et sw. Par la suite, on initie les coordonnées du snake avec snk_x et snk_y qui aideront à définir le corps de ce serpent (snake). On fait de même pour la nourriture.

Enfin on réalise une boucle pour capter les directions du serpent (getch) et s'il heurte le rebord de la fenêtre alors le jeu se réinitialise (endwin // quit).

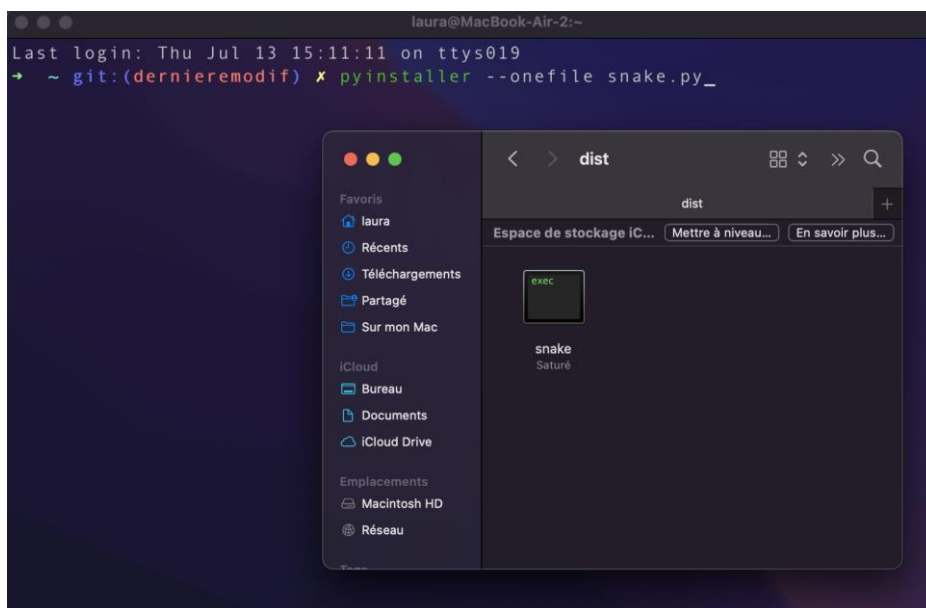
La nourriture retrouve une nouvelle position une fois que le serpent la "mange".

Pour le rendre exécutable, j'ai installé *Pyinstaller* avec **Pip** qui est un gestionnaire de paquets pour Python (-onefile permet d'éviter de recréer un fichier).

DOSSIER PROFESSIONNEL (DP)



Installation de Pip avec Python.



Usage de pyinstaller pour le snake.

2. Précisez les moyens utilisés :

- Python
- VS Code (ide)
- Pip (gestionnaire de paquets)
- Pyinstaller

3. Avec qui avez-vous travaillé ?

DOSSIER PROFESSIONNEL (DP)

4. Contexte

Nom de l'entreprise, organisme ou association ► LaPlateforme_

Chantier, atelier, service ► Snake Game

Période d'exercice ► Du : 2023-07-11 au : 2023-07-11

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.

DOSSIER PROFESSIONNEL ^(DP)

Déclaration sur l'honneur

Je soussigné(e) [prénom et nom] **Laura Savickaite** ,
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 **14/07/2023**

pour faire valoir ce que de droit.

Signature : [Laura SAVICKAITE](#)

DOSSIER PROFESSIONNEL ^(DP)

Documents illustrant la pratique professionnelle

(facultatif)

Intitulé
Cliquez ici pour taper du texte.

DOSSIER PROFESSIONNEL ^(DP)

ANNEXES

(Si le RC le prévoit)