

Titre: Développeur web et web mobile



Corentin Roussel

Dossier de projet

Table des matières:

Compétences du référentiels couvertes par le projet	3
Introduction	3
Spécifications fonctionnelles:	
Description de l'existant	4
Périmètre du projet	4
Cible adressée par le site internet	4
Arborescence du site	4
Description des fonctionnalités:	
1. Authentification	5
2. Catalogue produit et filtre	5
3. Fiche produit	5
4. Panier client	5
5. Fonctionnalité de recherche de produit	6
6. Espace client	6
7. Ajout d'adresse	6
8. Back office	6
8.1. Ajouter des catégories et sous-catégories de produits plateforme	6
8.2. Gestion des produits	6
8.3. Gestion des utilisateurs	7
9. Solution de paiement	7

Spécifications techniques:

Choix techniques et environnement de travail	7
Architecture du projet	8
Réalisations	
1. Charte graphique	9
2. Maquette	9
3. Conception de la base de données	10
4. Extraits de code significatifs	
4.1. Ajout d'article à l'aide du back office	10
4.2. Page index	13
4.3. Formulaire de paiement	
4.4. Carrousel JS	
5. Veille de vulnérabilités et de sécurités	
5.1. Faille XSS	
5.2. Faille d'upload	
5.3. Faille injection SQL	
6. Recherche effectué à partir d'un site anglophone	

Annexes

- Maquette
- Modèle conceptuel de données
- Modèle logique de données

Compétences du référentiel couvertes par le projet

Le projet couvre les compétences énoncées ci-dessous.

Pour l'activité 1, “**Développer la partie front-end d'une application web et web mobile en intégrant les recommandations de sécurité**”:

- Maquetter une application
- Réaliser une interface utilisateur web ou mobile statique et adaptable
- Développer une interface utilisateur web dynamique
- Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce

Pour l'activité 2, “**Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité**”:

- Créer une base de données
- Développer les composants d'accès aux données
- Développer la partie back-end d'une application web ou web mobile
- Élaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce.

Introduction

Boutique en ligne est le projet de fin d'année, il consiste à créer avec un groupe de 3 personnes sur une durée de 1 mois une boutique en ligne en utilisant les langages PHP, JavaScript, SQL.

Nous avons dû maquetter l'application à l'aide du logiciel figma et nous avons utilisé le logiciel Lucidchart pour la conception de la base de données.

Le site permettra à l'utilisateur de s'inscrire et de se connecter de pouvoir naviguer sur le site que ça soit sur la page index avec les jeux en précommande les jeux qui viennent de sortir etc... cherchez un jeu via la barre d'auto complétion de naviguer aussi sur la page tous les produits et de filtrer afin d'affiner la recherche et d'aller sur la page de produit rajouter un produit au panier, pouvoir consulter son panier et enfin pouvoir procéder au paiement, aussi aller sur la page de profil afin de pouvoir modifier ces informations personnelles ou compléter son profil tel que la date de naissance, le nom, le prénom.

Composé aussi d'un back-office qui permettra à l'administrateur de gérer les droits des utilisateurs, gérer les produits, gérer les catégories et sous-catégories.

Spécifications fonctionnelles

Description de l'existant

Il n'y avait pas de boutique nous avons dû tout créer de A à Z du maquettage de l'application en passant par la conception de la base données mais aussi l'entièreté des fonctionnalités codé par nos soins.

Périmètre du projets

Le site sera réalisé en anglais et ce dernier devra être accessible sur différents supports, à savoir mobile, tablette et ordinateur.

Cible adressée par le site internet

Le site que nous avons créé Game Vault cible une clientèle de joueur principalement console avec des jeux en ventes physiques

Arborescence du site

L'arborescence du site est définie telle qu'elle :

- Page d'accueil
- Page d'authentification (Connexion, Inscription)
- Page tous les produits
- Page produit
- Page panier
- Page coordonnée
- Page paiement
- Page récapitulative d'achat
- Page profil
- Page d'administration

Description des fonctionnalités

1. Authentification

La page d'authentification nous permet grâce à l'api fetch et javascript de récupérer les formulaires sur une même page, ce qui nous permet en fonction du bouton sur lequel on clique de récupérer le formulaire en question et de concentrer l'inscription et la connexion sur la même page a noter que a l'inscription un panier est créée automatiquement afin que l'utilisateur est un panier disponible tout le temps.

2. Catalogue produit et filtre

Le catalogue de produit doit être représenté sur une page qui contient tous les jeux, afficher tous les produits disponibles, avec la photo du produit, le nom du produit ainsi que le prix du produit et les filtres qui vont permettre d'affiner la recherche par console catégorie de jeux et sous-catégories.

3. Fiche produit

La fiche de produit va permettre de récupérer les détails du jeu en question, il va contenir :

- Synopsis du jeu
- La photo du jeu
- Le titre du jeu
- Détails de l'éditeur et du producteur
- La date de sortie
- Récapitulatif des catégories
- Choix de la plateforme
- Quantité de jeux

4. Panier client

Le panier va venir récupérer tous les jeux que l'on a ajoutés dedans, il va permettre de gérer ses achats en supprimant un jeu si on en voit plus l'utilité, ou ajouter un exemplaire du jeu que nous achetons ou enlever un exemplaire si on s'est trompé de quantité par exemple, il va aussi permettre d'obtenir un récapitulatif qui contiendra :

- L'image de l'article
- Le titre de l'article
- La plateforme choisie
- La quantité choisie
- Le prix total du panier
- Un bouton de validation de panier

5. Fonctionnalité recherche de produit

La fonctionnalité recherche de produit va permettre avec une barre d'autocomplétion de rechercher un jeu plus précisément, on a juste à taper le jeu que l'on essaye de trouver dans la barre ce qui va nous sortir tous les jeux qui ont des lettres similaires par exemple si nous voulons rechercher Zelda, il suffit de taper Zelda afin de pouvoir voir tous les jeux Zelda apparaître.

6. Espace client

L'espace client va permettre aux clients inscrits sur le site de modifier leurs informations personnelles ou d'ajouter des informations personnelles telles que le numéro de téléphone, nom, prénom, date de naissance, email, mais aussi de pouvoir voir l'historique de ces achats.

7. Ajout d'adresse

L'ajout d'adresse va permettre d'attribuer plusieurs adresses à une personne elles seront stockés directement en base de données à l'aide d'un formulaire qui sera prêt rempli si l'utilisateur a rempli ces informations via le profil, les stockés en base de données va nous permettre de pouvoir les retrouver lors de la simulation de paiement à l'aide de l'id utilisateur actuellement connecté.

8. Back office

8.1. Ajouter des catégories sous-catégories de produits et plateforme

La problématique était que si une nouvelle catégorie ou sous catégories de jeu était créée, on puisse la rajouter et l'utiliser de façon à ce que l'on ai pas besoin de le rajouter dans la base de données pour l'ajout des plateformes, c'est le même principe que pour les catégories si une nouvelle console sort, on peut l'ajouter.

8.2. Gestion des produits

Le back-office de gestion de produits va permettre de rajouter des jeux au fur et à mesure ou de supprimer des jeux qui ne sont plus à la vente qui permet aussi de modifier des produits suivants si une erreur a été commise ou non.

8.3. Gestion des utilisateurs

La gestion des utilisateurs va permettre de gérer les droits des utilisateurs ou de les supprimer.

9. Solution de paiement

La solution de paiement va permettre de finaliser la commande en simulant un paiement réel, en vidant le panier existant et en créant un nouveau afin que l'utilisateur et toujours un panier disponible, elle vous permettra de sélectionner l'adresse à laquelle vous voudrez vous faire livrer et aurez un récapitulatif des achats effectués.

Spécifications techniques

Choix techniques et environnement de travail

Technologies utilisées pour la partie front-end :

- Le HTML pour le display des éléments de la page HTML
- Le CSS pour donner du style et rendre le site responsive
- Le JavaScript pour le dynamisme et l'interactivité de la page et d'améliorer l'expérience utilisateur

Technologies utilisées pour la partie back-end :

- Le langage PHP (Hypertext Processor) sera utilisé principalement pour le back avec une POO (Programmation Orienté Objet)
- Utilisation d'un base données SQL (Structured Query Language) avec PDO (PHP Data Objects)

L'environnement de développement :

- Éditeur de code: PHP Storm.
- Outil de versioning: Git, GitHub.
- Maquettage: Figma.
- Conception de base de données: Lucidchart

Du côté de l'organisation nous n'avons pas utilisé de méthode de travail en particulier, nous avions un chat google et un chat discord qui permettait de discuter de l'avancement du projet et de prévenir les autres membres du groupe d'un changement, d'une modification ou de la complétion d'une tâche, un membre de l'équipe a été assigné dès le début du projet afin d'avoir une personne plus ou moins en charge des décisions finales, nous avons aussi également des table ronde en fin de semaine généralement afin de déterminer les choses à

finaliser en priorités si c'était à refaire, je pense que l'utilisation d'un outil de gestion tel que trello aurait pu être bénéfique pour ce projet.

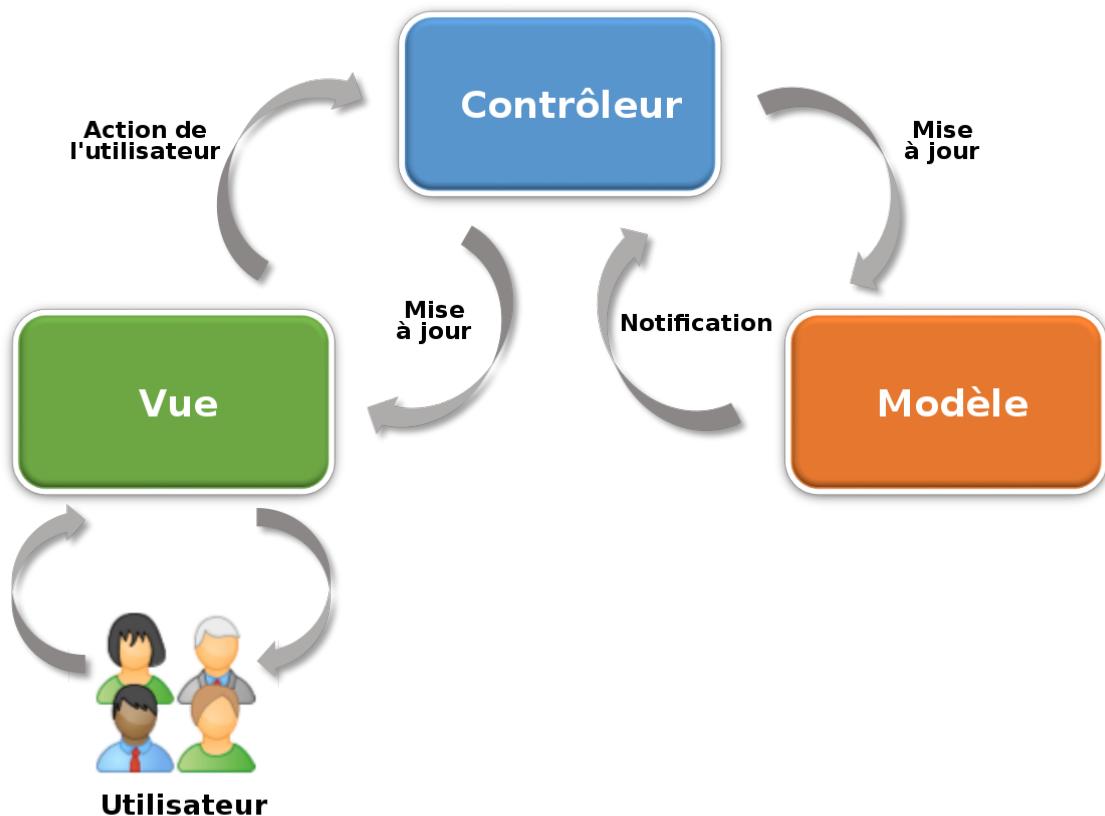
Architecture de projet

Le projet est développé à l'aide d'un design pattern de type MVC (Model - View - Controller).

L'architecture MVC et l'une des architectures les plus utilisées pour les applications Web, elle se compose en 3 modules :

- **Modèle** : noyau de l'application qui gère les données, permet de récupérer les informations dans la base de données, de les organiser pour qu'elles puissent ensuite être traitées par le contrôleur.
- **Vue** : Composant graphique de l'interface qui permet de présenter les données du modèle à l'utilisateur.
- **Contrôleur** : Composant responsable des prises de décision, gère la logique du code, il est l'intermédiaire entre le modèle et la vue.

Source : Wikipédia



Réalisations

1. Charte graphique

La charte graphique va permettre de déterminer en avance les couleurs que l'on va utiliser étant donné que c'est un projet de groupe établir les couleurs utilisées tout au long du projet va permettre de faciliter la conception du style de la boutique en ligne et toutes les personnes travaillant sur le projet savent quelles couleurs utiliser

Police :
Montserrat

Couleur :



2. Maquette

La maquette a été réalisée avec figma, n'ayant pas de designer sur le projet nous avons pris une grande inspiration sur le site Instant Gaming qui nous a permis de nous lancer directement dans la conception du figma sans avoir besoin de perdre trop de temps à choisir grandement et d'avoir un design propre et le but principal était de reproduire le site de la manière la plus fidèle la maquette et le wireframe seront disponible à la fin du dossier.

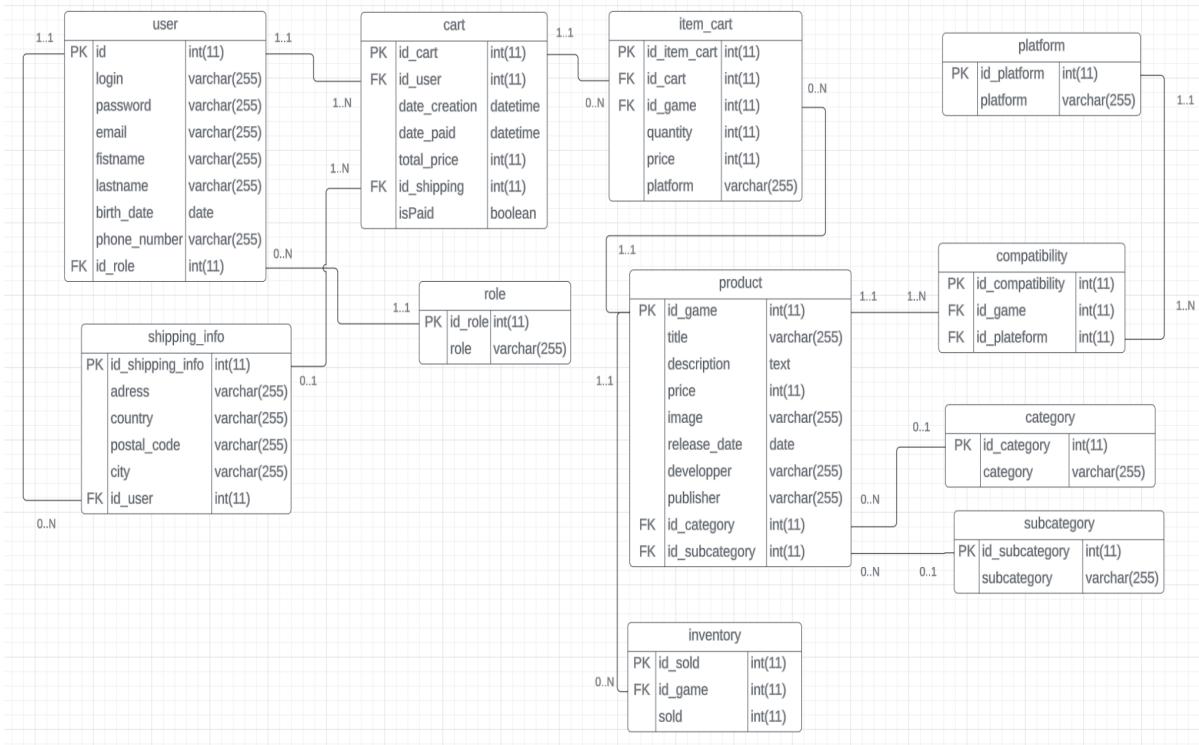
On a du commencer par prototyper le site à l'aide d'un wireframe, le wireframe c'est une ébauche du site qui va permettre d'avoir une vue d'ensemble sur ce qui va composer le site c'est tout d'abord créer uniquement avec des nuances de gris pour ne pas influencer le design pure avec des couleurs on a aussi tous les composants qui vont faire le site.

La maquette va permettre d'avoir un visuel du site tel qu'on le voudrait sur la version du site final, il va permettre d'utiliser les couleurs de la charte graphique pour pouvoir donner le style de couleurs à la page comme ça tout le monde se réfère à ce dernier et ce qui permet un gain de temps énorme.

3. Conception de base de données

Le MCD (modèle conceptuel de données) et MLD (modèle logique de données) sont disponibles à la fin du dossier.

Voici donc la base de données (MPD) conçu pour le projet :



La base de données ci-dessus nous permet de visualiser que la table user va permettre d'enregistrer et de connecter nos utilisateurs. La table user est liée à la table shipping_info ce qui va permettre à l'utilisateur de cumuler plusieurs adresses aussi liées a cart ce qui va permettre de lier un seul panier à la fois à un utilisateur.

Il y a la table product qui elle est lié à la table de liaison compatibility qui est elle même relié à la table platform une table de liaison c'est quoi et bien ça permet de donner une platform a plusieurs jeux et qu'un jeux peut bénéficier de plus d'une platform mais elle est aussi lié aux tables catégories et subcategory afin de pouvoir leur assigner des étiquettes pour rendre la recherche des jeux plus simple et la table inventory qui est lié aux jeux afin de pouvoir retrouver le nombre de jeux vendus et peut être plus tard gérer le stock et enfin la table item_cart qui est la table de liaison entre product et cart qui va permettre de stocker tout les produits dans un cart.

4. Extraits de codes significatifs

4.1. Ajouts d'article à l'aide du back office

L'ajout de l'article et la base de la boutique en ligne, c'est la fonction qui va nous permettre de remplir la boutique du produit qui est vendu, j'ai tout d'abord créé un formulaire HTML qui va me permettre de renseigner toutes les informations nécessaires pour représenter un jeu. J'ai donc avec ce formulaire dû établir quelques méthodes me permettant de :

- Insert Game

```
public function insertGame($title, $desc, $price, $image, $date, $developper, $publisher, $checkboxArray, $category, $sub_category):void
{
    $messages = [];

    $priceInt = (int)$price;
    $categoryInt= (int)$category;
    $sub_categoryInt= (int)$sub_category;

    $checkTitle = $this->model->checkIfTitleIsSet($title);

    if(preg_match("#^[0-9]*$#", $price) && grapheme_strlen($desc) > 100 && $this->mempty($title, $desc, $price, $image, $date, $publisher, $developper, $category, $sub_category) && $checkTitle === 0 && $checkboxArray !== null)
    {
        $this->model->reqInsertGame($title, $desc, $priceInt, $image, $date, $developper, $publisher, $categoryInt, $sub_categoryInt);

        $this->setPlatform($checkboxArray);

        $messages[ 'okAddGame' ] = "You're game has been added to the product list";
    }
}
```

Pour insérer un jeu en base de donnée on va d'abord récupérer les éléments du formulaire en POST je viens passer le pris et les catégories en int puisque les valeurs passer en post sont forcément transformer en string avec une autre méthode je regarde si le titre du jeu n'est pas déjà pris afin d'éviter les doublons et après les vérifications que le prix est bien un int que la description fait plus de 100 caractères, qu'aucun des champs ne sont vides, que le titre soit bien disponible et que au moins une checkbox pour la plateforme a bien était coché on insère le jeu en base de données et on vient aussi setPlatform car un jeux peut avoir plusieur plateforme.

- Update Game

```
$this->model->updateById($title, $desc, $priceInt, $image, $date, $developper, $publisher, $categoryInt, $sub_categoryInt,$id);

$this->model->deleteCompat($id);
$this->updatePlatform($checkboxArray, $id);
```

L'update game est plus ou moins la même que l'insert sauf que pour les plateformes qui sont rangées dans une table de liaison, j'ai dû m'adapter pour pouvoir changer les plateformes pour le jeu déjà créé, il fallait que je delete les plateformes dans cette table de liaison et que je les insère de nouveau.

- Checkbox plateforme

Voici donc comment je récupère les plateformes de jeux et ensuite j'utilise un name check_list qui va regrouper les plateformes sous forme de tableau que l'on va ensuite envoyer en post, on va vérifier comme au dessus que checkbox n'est pas null autrement dit que une valeur a bien était coché et ensuite je le dernier jeux inséré pour récupérer son id et pour chaque checkbox coché je viens rempli la table compatibility

```
public function setPlatform(array $arrayCheckbox):void {  
    $id_game = $this->model->fetchLastGame();  
  
    foreach ($arrayCheckbox as $key => $platform){  
        $intPlatform = (int)$platform;  
  
        $this->model->insertPlatform($id_game['id'], $intPlatform);  
    }  
}
```

```
//name dans l'input type=checkbox  
name="check_list[]"  
//champ post récupère qui va nous donner  
//un tableau de chaque bouton avec le name check_list  
$_POST['check_list']
```

- Verif champs vides



```
public function security(array $array):array | false
{
    $result = [];
    foreach ($array as $key => $values)
    {
        $result["$key"] = htmlspecialchars(trim($values));

        if(!empty($values))
        {
            continue;
        }else {
            return false;
        }
    }
    return $result;
}
```

Pour la vérification des champs vide j'ai créé, une fonction afin de ne pas avoir à faire !empty(variable) j'ai donc créer une fonction qui check si les champs sont vides mais qui htmlspecialchars et trim aussi toute les values reçu on peut donc voir que l'on passe un tableau à cette fonction on crée un nouveau tableau vide pour chaque valeur on va htmlspecialchars et trim la nouvelle valeur et ensuite on vérifier que cette valeur n'est pas vide et ensuite on return \$result cette fonction nous renvoie bien soit un array soit false

4.2. Page index

La page, index va permettre de display suivant notre maquette un carrousel qui va contenir les précommandes de jeux ensuite un scroll horizontal qui présente les jeux qui viennent de sortir, douze jeux aléatoirement générés et ensuite les jeux les plus vendus sur le site. Allons voir comment c'est fait :

- Affichage de jeux

```
public function getBestSellerGames():array {
    foreach($this->model->bestsellersGames() as $games)
    {
        $game_price = substr_replace($games['price'], ".", -2, 0) . "€";
        $game[] = '<div class="new-released-game">
                    <a href="product.php?id=' . $games['id_game'] . '"></a>
                    <div class="new-released-games-title-price">
                        <a href="product.php?id=' . $games['id_game'] . '" class="link-released-games"><p class="new-released-text">' . $games['title'] . '</p></a>
                        <p class="new-released-price">' . $game_price . '</p>
                    </div>
                </div>';
    }
    return $game;
}
```

Pour l'affichage de jeux je récupère avec une requête que vous verrez ci-dessous, je boucle sur les résultats de ma requête avec un foreach je change le prix qui est stocké en int dans la base de données et applique une virgule ensuite je crée un tableau vide afin de pouvoir push la string pour chaque jeux qui va me permettre d'afficher tous les jeux avec le bon format et ensuite je return le tableau remplis de tout les jeux.

- Jeux en précommandes

```
public function preorderGames(): array
{
    $req = $this->conn->prepare("SELECT
                                product.id,
                                product.title,
                                product.price,
                                product.image,
                                product.release_date
                            FROM product
                            WHERE release_date BETWEEN CURDATE()
                            AND DATE_SUB(CURDATE(), INTERVAL -6 MONTH)
                            ORDER BY product.release_date DESC");
    $req->execute([]);
    echo json_encode($req->fetchAll(PDO::FETCH_ASSOC), JSON_PRETTY_PRINT);
    die();
}
```

Ici, on peut donc voir la requête qui permet d'alimenter l'affichage de jeux, je sélectionne tout ce dont j'ai besoin et je n'utilise pas l'étoile ce qui augmente les performances de la requête j'utilise aussi BETWEEN CURDATE() pour chercher entre deux dates distinctes afin de récupérer dans ce cas-là les jeux qui ne sont pas encore sorties avec un intervalle de six mois.

- Jeux les plus vendus

```
public function bestsellersGames(): array
{
    $req = $this->conn->prepare("SELECT inventory.sold,
                                inventory.id_game,
                                product.title,
                                product.image,
                                product.price FROM inventory
                                INNER JOIN product
                                ON product.id = inventory.id_game
                                ORDER BY inventory.sold DESC LIMIT 5");
    $req->execute();
    return $req->fetchAll(PDO::FETCH_ASSOC);
}
```

Pour les jeux les plus vendus, je me suis rendu compte qu'il fallait que je fasse une table inventory afin d'avoir toutes les ventes à chaque fois qu'un panier est validé, je récupère la quantité du jeu en question et l'insère dans la base de données et utilise cette requête pour pouvoir les récupérer.

- Jeux qui viennent de sortir

```
public function newReleasedGames(): array
{
    $req = $this->conn->prepare("SELECT
                                product.id ,
                                product.title,
                                product.price,
                                product.image
                                FROM product WHERE release_date
                                BETWEEN DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
                                AND CURDATE() ORDER BY product.release_date DESC LIMIT 5");
    $req->execute([]);
    return $req->fetchAll(PDO::FETCH_ASSOC);
}
```

Pour les jeux qui viennent de sortir, c'est pratiquement la même chose que les jeux en précommandent sauf qu'au lieu des jeux qui ne sont pas encore sortis, j'utilise la même fonction pour trouver les produits qui viennent de sortir dans les 3 mois.

4.3. Formulaire de paiement

Le formulaire de paiement est une simulation de paiement, il va permettre de valider son panier de choisir son adresse ainsi que les informations de carte bancaire nécessaire afin de pouvoir payer son panier et d'en récupérer un nouveau.

Je vous explique comment on fait :

- Simulation du paiement

```
public function verifFormPayment(?int $idAdress, ?string $cardNumber, ?string $cardExpiration, ?string $cardAuth,
?string $cardName, ?int $id_cart, ?int $id_user) {

    $cardNumberRep = str_replace(" ", "", $cardNumber);
    $cardExpirationRep = str_replace("/", "", $cardExpiration);
    $date = date("Y-m-d H:i:s");
    $messages = [];

    if(is_numeric($idAdress) && ctype_digit($cardNumberRep) && strlen((string)$cardNumberRep) == 16 &&
    ctype_digit($cardExpirationRep) && grapheme_strlen($cardExpirationRep) == 4 && ctype_digit($cardAuth) &&
    grapheme_strlen((string)$cardAuth) == 3 && isset($cardName) && isset($messages))
    {
        }
}
```

Pour la simulation du paiement j'ai du remplacer pour le numéro de carte bleue et le date d'expiration les espaces potentielles des personnes et les slash, j'ai aussi récupéré la date afin de savoir quelle jour et à quelle heure le panier a t-il été validé je vérifie aussi que toute les entrées sont bien des nombres, is_numeric vérifie que la variable est un nombre ou une chaîne numérique et ctype_digit vérifie que la chaîne passé est bien un entier et je vérifie aussi que le nombre de caractères rentré est le bon

- Gérer le panier

```
if(is_numeric($idAdress) && ctype_digit($cardNumberRep) && strlen((string)$cardNumberRep) == 16 &&
ctype_digit($cardExpirationRep) && grapheme_strlen($cardExpirationRep) == 4 && ctype_digit($cardAuth) &&
grapheme_strlen((string)$cardAuth) == 3 && isset($cardName) && isset($messages))
{
    $games_cart = $this->model->getItemCart($id_cart);
    $this->model->insertSoldGames($games_cart);
    $this->model->cartBought($idAdress, $date, $id_cart);
    $this->model->setCart($id_user, $date);
    $id_actual_cart = $this->model->getCart($id_user);
    $_SESSION['user']['actualCart'] = $id_actual_cart['id'];
    $messages['okCart'] = "okCart";
}
```

Maintenant que toutes les conditions ont été remplies on va pouvoir utiliser les méthodes créés au préalable on va devoir récupérer les produits du panier avec getItemCart afin de pouvoir récupérer les jeux la quantité de jeux vendus et leur prix et l'insérer avec insertSoldGames on va vérifier si le jeu que l'on ajoute existe déjà ou non dans la table inventory on utilisera UPDATE ou INSERT INTO suivant le cas de figure, ensuite le panier a été acheté la méthode cartBought se déclenche avec l'adresse de livraison la date à laquelle il a été payé et l'id du cart en question on va venir utiliser setCart pour pouvoir réinitialiser un nouveau panier et que l'utilisateur est toujours un panier actif et ensuite on

récupère le panier qui vient d'être créé et on remplace l'ancien panier en session avec le nouveau panier.

4.4. Carrousel JS

Pour pouvoir display les jeux en précommandes, j'ai décidé de faire un carrousel j'ai, donc utilisé JS:



```
const displayImage = (e, array, previous, next, link, img) => {
    let i = 0;

    link.href = "product.php?id=" + array[i].id;
    img.src = array[i].image;

    next.addEventListener("click", () => {
        i++;

        if (i > array.length - 1) {
            i = 0;
        }

        link.href = "product.php?id=" + array[i].id;
        img.src = array[i].image;
    });

    previous.addEventListener("click", () => {
        i--;

        if (i < 0) {
            i = array.length - 1;
        }
        link.href = "product.php?id=" + array[i].id;
        img.src = array[i].image;
    });
};
```

Cette fonction va prendre en paramètre un événement un array previous et next qui correspondent à des boutons, et le lien et l'image du carrousel on va lui assigner un lien et une image de base avec le premier index du tableau ensuite on utilise un addEventListener pour écouter l'événement au click sur le bouton next pour pouvoir incrémenter la variable i pour pouvoir changer de lien et d'image on établit aussi une condition qui dit que si on arrive à la fin de l'array on reset la variable i à 0 pour revenir au début on reset du coup le lien et l'img avec le changement de la variable i, on utilise la même logique pour le bouton previous mais on décrémente et si on arrive au début de l'array si on clique sur previous on reset la variable i pour quelle revienne à la fin



```
const getArray = async () => {
    const response = await fetch("index.php?getArray=ok");
    return await response.json();
};

window.addEventListener("load", async (e) => {
    let array = await getArray();
    displayImage(e, array, previous, next, link, img);
});
```

Voici du coup la méthode fetch qui récupère les images et liens des jeux en précommandes et les formate en JSON et ensuite dès que la page se charge, on récupère du coup les images et liens et on les display a l'aide de la fonction displayImage expliqué ci-dessus.

Veille de vulnérabilités sécurités

4.5. Faille XSS

La faille XSS (Cross-Site Scripting) est une vulnérabilité particulièrement répandue dans les applications web. Les XSS font partie de la catégorie des **vulnérabilités par injection de code** au même titre que les injections SQL que l'on verra plus tard. L'encodage (ou échappement) des données en entrée ou en sortie reste la mesure de sécurité essentielle pour prévenir les attaques XSS. L'encodage d'entités HTML est sûrement la mesure la plus essentielle car les scripts malveillants sont souvent injectés via des balises HTML. Ici, il s'agit donc d'encoder la plupart des entités HTML pouvant constituer un risque afin qu'elles soient interprétées comme des données fiables. Par exemple, l'entité « < » , normalement interprétée comme le début d'une balise HTML pourrait être encodée en « < » de manière à ce qu'elle soit reçue et traitée par l'application comme telle. De la même manière, il est possible (et fortement recommandé) d'encoder les valeurs d'attributs qui peuvent être utilisées pour injecter des scripts malveillants comme : href, src, style, onerror, etc.

4.6. Faille d'upload

La **faille upload** est une faille permettant d'uploader des fichiers avec une extension non autorisée, cette faille est due à la mauvaise configuration du script d'upload ou à l'absence complète de sécurité. Celle-ci est généralement présente dans les scripts d'upload d'images. C'est une des failles les plus dangereuses. Le but de cette faille est d'uploader un fichier avec une extension non autorisée. (Par exemple un code php) de façon à avoir un accès au serveur cible. Si le formulaire d'upload de votre site n'est pas sécurisé, alors un pirate informatique pourrait sans problème s'amuser à uploader un fichier PHP malveillant (web shell par exemple) qui lui permettrait de prendre le contrôle total de votre application web, et de votre serveur.

4.7. Faille d'injection SQL

Comme son nom l'indique, **l'injection SQL (ou SQLi)** est une méthode d'exploitation de faille de sécurité d'une application possédant des interactions avec une base de données. Le principe est d'injecter, dans une requête, du code SQL malveillant qui viendra modifier l'effet escompté et ainsi compromettre l'intégrité des données présentes dans la base. Cette technique est très souvent utilisée pour contourner les mécanismes d'authentification et d'autorisation d'une application web.

Une faille SQLi peut avoir de lourdes conséquences car un hacker peut avoir un accès non autorisé aux données sensibles. Il sera en mesure de lire la base de données, y enregistrer de nouvelles données ou exécuter du code malveillant. Quand on connaît la valeur et l'importance des données, il serait dommageable pour un site de subir une telle cyberattaque.

Afin de se protéger contre les failles SQLi, il est préférable d'utiliser un système de requêtes préparées (PDO pour PHP par exemple). La requête est compilée avant d'être exécutée pour s'assurer qu'elle ne contient pas de caractères d'échappement.

5. Recherche à partir de site anglophone

Stack overflow: Comment poster plusieurs <input type="checkbox" /> en tableau dans PHP

[https://stackoverflow.com/questions/1978781/how-to-post-multiple-input-type-checkbox-as-a](https://stackoverflow.com/questions/1978781/how-to-post-multiple-input-type-checkbox-as-an-array-in-php)
[array-in-php](#)

La personne posant la question voulait récupérer plusieurs input checkbox sous forme de tableau afin de pouvoir utiliser une boucle foreach afin de savoir lequel est check ou pas

```
foreach($_POST['checkboxname'] as $i => $value)
```

La réponse lui dis de procéder comme ceci

```
<input type="checkbox" name="checkboxArray[]" />
```

attention au tableau dans le nom.

Utiliser if(!empty) avec plusieurs variables pas dans un tableau

<https://stackoverflow.com/questions/4993104/using-ifempty-with-multiple-variables-not-in-an-array>

j'essaye de refactoriser un peu mon code avec le if(!empty) fonction en PHP mais je ne sais pas comment appliquer cela à plusieurs variables quand ce n'est pas un tableau (comme je devais le faire avant) donc si j'ai:

```
$vFoo = $item[1];  
$vSomeValue = $item[2];  
$vAnother = $item[3];
```

Après je veux imprimer le résultat seulement s'il y'a une valeur. Cela marche avec une valeur donc on a:

```
if (!empty($vFoo)) { $result .= "<li>$vFoo</li>"; }
```

J'ai essayé quelque chose comme ça:

```
if(!empty($vFoo,$vSomeValue,$vAnother)) { $result .= "<li>$vFoo</li>" $result .=  
"<li>$vSomeValue</li>" $result .= "<li>$vAnother</li>" }
```

Mais bien sur ça ne fonctionne pas

Tu pourrais faire une nouvelle fonction qui accepte plusieurs arguments et passe à travers a chaque fois par empty(). Ca marchera pareil que isset(), qui retournera true seulement si tous les arguments ne sont pas vide, et retournera false dès qu'il rencontrera le premier argument qui n'est pas vide. Voilà ce que j'ai réussi à faire, et ça marche dès que je l'ai testé.

```
function mempty() {  
    foreach(func_get_args() as $arg)  
        if(empty($arg))  
            continue;  
    else  
        return false;  
    return true;  
}
```

Pour plus de précision: Le premier m dans "mempty" est la pour "multiple". Vous pouvez l'appeler comme vous voulez, mais ça me semblait la façon la plus courte/facile de l'appeler. En plus c'est amusant à dire.

Update 10/10/13: Je devrais probablement ajouter que non comme empty() ou isset(), cette fonction mempty() va crier au meurtre si vous lui passez une variable non-défini ou un index de tableau qui n'existe pas.

Annexes

Wireframe

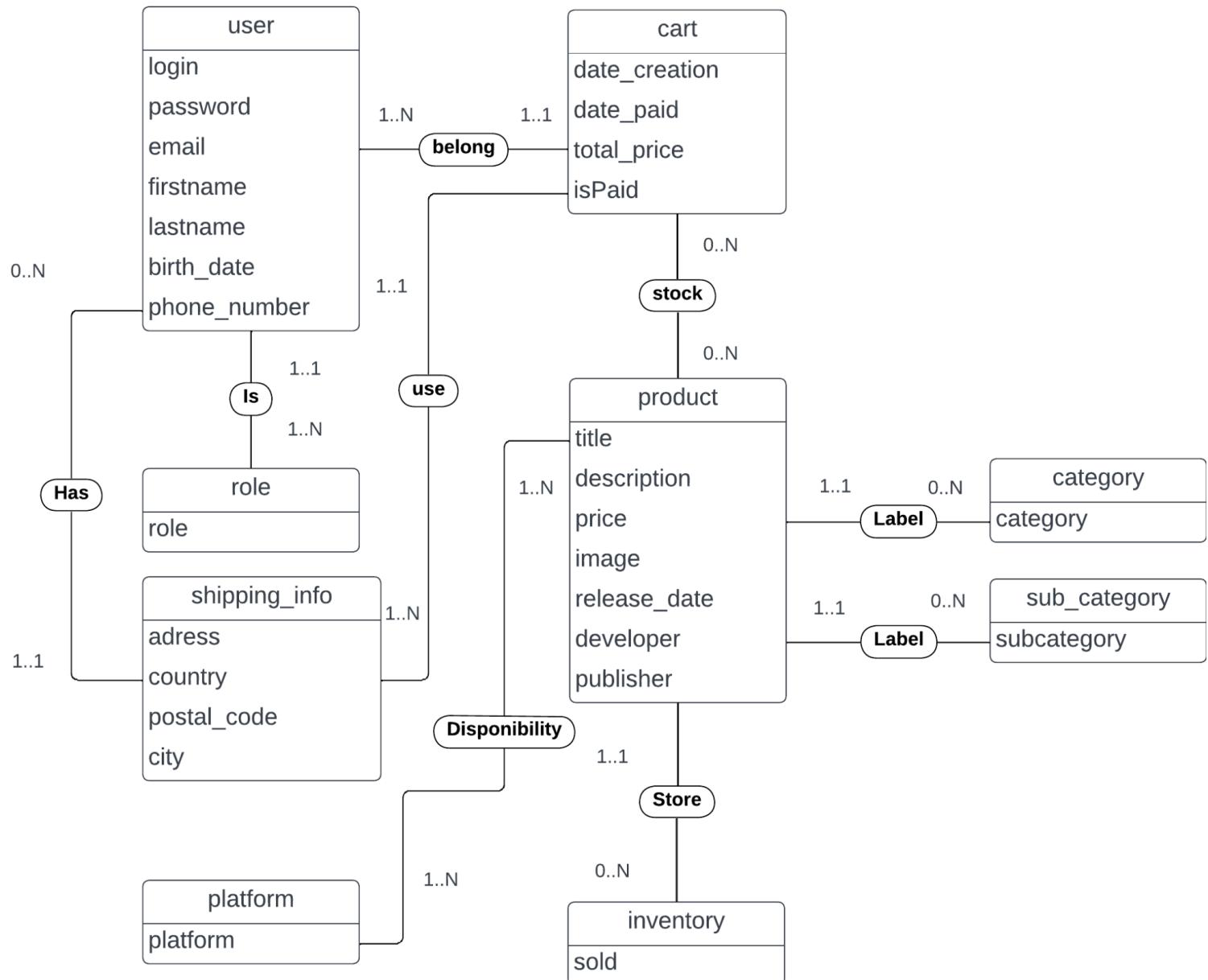


Maquettes

The image displays six wireframe mockups of a game store website, likely for a mobile application, arranged in a 2x3 grid. The website is branded as 'GAMER VAULT'.

- Top Left:** All products (Desktop). Shows a grid of game thumbnails like 'Police Craft', 'DREDGE', 'Resident Evil 4', etc., with filters for Platform, Category, and Sub category.
- Top Middle:** All products desktop. Similar to the top-left, but with a different layout and some UI elements.
- Top Right:** Desktop - 4. Shows an order summary for Order #001, listing 'Resident Evil 4' (PS5) and 'The Last Of Us part 1' (PS5), with a total of 0,00€.
- Bottom Left:** Cart. Shows a cart with a message: "Your cart is empty. You have added nothing into your cart. Please browse our site to find incredible offer!"
- Bottom Middle:** Cart empty-desktop. Similar to the bottom-left, but with a different UI for the empty cart message.
- Bottom Right:** Cart-filled-desktop. Shows a cart filled with 'Resident Evil 4' and 'The Last of us', with a summary price of 0,00€ and a "Go to payment" button.

MCD



MLD

