UNIVERSITÉ DE LORRAINE | IUT Metz

SAÉ 4.01

Livrable 1 - Analyse

BONDON Loric

BROCHOT Joshua

BRODIER Baptiste

ROTH Sevan

Année 2024/2025

I. Clone d'application

Le clone de l'application se situe sur :

GitHub via ce lien :
https://github.com/LoricBndn/SAE4.01

DevWeb via ce lien :
https://devweb.iutmetz.univ-lorraine.fr/~bondon3u/2A/SAE4.01/client/accueil.html

PhpMyAdmin pour la BDD.

II. Application mobile Android

L'application mobile Android (fichier APK) via Cordova se situe sur :

GitHub via ce lien :
https://github.com/LoricBndn/SAE4.01

III. Analyse complète de l'application

A. Analyse technique

1. Architecture

Problèmes liés à l'arborescence du dossier client

a) Côté client

Actuellement, les fichiers HTML sont placés directement dans client/, ce qui complique la gestion et la lisibilité du projet.

Problème : Absence de structure claire pour les fichiers HTML.
Solution : Déplacer ces fichiers dans un sous-dossier client/pages/ pour mieux organiser les différentes vues.

Le dossier controleur/ contient des fichiers JavaScript mal nommés ou mal organisés.

Problème :

- function.js est trop générique et devrait être renommé en utils.js pour être plus explicite.
- regionApi.js, qui semble spécifique aux régions, est placé parmi des scripts de logique générale.

Solution : Séparer les scripts en sous-dossiers selon leur rôle :

client/js/

controllers/ -> accueil.js, compte.js, login.js, panier.js, etc.

utils/-> utils.js, regionApi.js, etc.

Le projet contient un fichier bootstrap.min.css, mais il n'est pas clair s'il est téléchargé ou utilisé via un CDN.

Problème : Manque de distinction entre ressources internes et externes.
Solution : Si bootstrap.min.css est téléchargé, il devrait être déplacé dans client/vendor/bootstrap/ pour éviter la confusion.

Le fichier style.css est directement placé dans css/, ce qui ne favorise pas une gestion modulaire des styles.

Problème : Fichier CSS unique difficile à maintenir.
Solution : Diviser les styles en plusieurs fichiers, par exemple :

client/css/

layout.css -> Structure générale

components.css -> Boutons, formulaires, etc.

theme.css -> Couleurs et typographie

Les images sont mélangées dans img/ et img/icones/ sans logique apparente.

Problème : Classement des fichiers peu intuitif.
Solution : Organiser les images en sous-dossiers selon leur utilisation :

client/img/

products/ -> Images des produits

icons/ -> Icônes de l'interface

backgrounds/ -> Fonds d'écran ou textures

L'arborescence actuelle ne permet pas de séparer clairement le code du frontend et du backend.

Problème : Le serveur et le client sont mélangés dans la structure du projet.
Solution : Ajouter un dossier server/ distinct pour la partie backend afin d'éviter toute confusion avec client/.

b) Coté serveur

Tous les fichiers API sont situés dans serveur/api/ sans structure claire, ce qui complique la navigation.

Problème : Trop de fichiers API au même niveau.
Solution : Réorganiser les fichiers en sous-dossiers selon leur fonction :

serveur/api/

auth/ -> connexion.php, changerMDP.php, newUser.php, delUser.php, getUser.php, getUsers.php

panier/ -> getPanier.php, setPanier.php, clearPanier.php, delPanier.php

produits/ -> getProduit.php, getProduits.php, newProduit.php, delProduit.php, setProduit.php

commandes/ -> getCommande.php, getCommandes.php, newCommande.php, payer.php

De plus, le fichier header.php, actuellement dans api/, devrait être déplacé dans serveur/includes/ pour être utilisé de manière globale.

Le fichier laroche5_pm2.sql est stocké directement dans serveur/.

Problème : Absence d'un dossier dédié aux bases de données.
Solution : Déplacer laroche5_pm2.sql dans serveur/database/.

Par ailleurs, chaque fichier PHP semble interagir directement avec la base de données sans centralisation.

Problème : Manque de gestion structurée des requêtes SQL.
Solution : Ajouter un fichier serveur/database/db.php pour gérer la connexion à la base de données et centraliser les requêtes SQL.

Le fichier index.php est placé directement dans serveur/, ce qui ne clarifie pas son rôle.

Problème : Absence de distinction entre fichiers internes et fichiers accessibles publiquement.
Solution : Si index.php est un point d'entrée principal, il devrait être déplacé dans serveur/public/.

2. Contenu de la partie Client

a) Vue (HTML)

Défaut	Description	Fichiers concernés
Manque d'éléments <meta> pour SEO et accessibilité	Certains fichiers ne contiennent pas de balises <meta name="description"> ou <meta name="keywords">, ce qui peut impacter le référencement.	`accueil.html`, `favori.html`, `panier.html`, `regions.html`
Utilisation de styles inline	Des styles CSS sont définis directement dans les fichiers HTML via style="", ce qui réduit la maintenabilité.	`panier.html`, `accueil.html`, `changer_mdp.html`
Balises <script> mal optimisées	Certains fichiers incluent des scripts avec type="module" deux fois sur la même ligne, ce qui peut causer des erreurs d'interprétation.	`historique.html`, `panier.html`
Utilisation de jQuery inutilement	jQuery est utilisé pour des manipulations DOM simples qui pourraient être faites avec du JavaScript natif (document.querySelector).	`accueil.html`, `changer_mdp.html`, `favori.html`, `register.html`
Importation de jQuery via CDN	Inclure jQuery via un CDN peut poser des problèmes de performance, de dépendance inutile et d'échec de chargement si la connexion est instable.	`accueil.html`, `changer_mdp.html`, `favori.html`, `register.html`, `login.html`, `panier.html`
Formulaires sans validation côté client	Plusieurs formulaires ne contiennent pas de validation JavaScript pour vérifier les champs avant l'envoi.	`register.html`, `login.html`, `changer_mdp.html`
Absence d'alternative textuelle (alt) pour les images	Certaines images ne possèdent pas l'attribut alt, ce qui impacte l'accessibilité.	`accueil.html`, `favori.html`, `regions.html`

Utilisation d'éléments pour la mise en page	Des sauts de ligne ( ) sont utilisés au lieu de marges CSS pour espacer les éléments.	`historique_detail.html, register.html, user.html`
Manque de gestion des erreurs dans les scripts	Certains scripts font des appels fetch() sans gestion des erreurs (catch).	`compte.html, favori.html, produit.html`

b) Style (CSS)

Défaut	Description	Fichiers concernés
Transitions globales trop larges	* { transition: 200ms; } applique une transition à tous les éléments, ce qui peut ralentir l'affichage et causer des effets visuels inattendus.	`*, *::before, *::after`
Images de fond en arrière-plan	background-image: url("../img/fondSite3.jpg"); est défini pour tout le site, ce qui peut poser des problèmes de performance et de lisibilité selon la taille de l'écran.	body
Suppression incomplète des boutons de spin pour input[type=number]	La gestion de -webkit-appearance: none; et -moz-appearance: textfield; est partielle et pourrait ne pas fonctionner sur tous les navigateurs.	`input::-webkit-outer-spin-button, input::-webkit-inner-spin-button, input[type=number]`
Problème de positionnement du logo	.logo { position: absolute; left: 1rem; } risque de mal s'adapter aux écrans plus petits.	`.logo`
Utilisation excessive des margin globales	body { margin: 0; padding: 0; width: 100%; } force la mise en page sans respecter les marges naturelles des navigateurs.	body

Flexbox mal géré pour le header	header nav { width: 30%; justify-content: right; } peut poser des problèmes sur les petits écrans.	`header nav`
Footer pas toujours bien positionné	footer { margin-top: auto; } fonctionne uniquement si le body a une hauteur suffisante (min-height: 100vh).	`footer`
Boutons trop peu visibles en hover	background-color: white; en hover sur .form_button, .btn_creation, .btn_retour peut réduire la lisibilité s'il n'y a pas de bordure.	`.form_button:hover, .btn_creation:hover, .btn_retour:hover`
Absence de styles pour h2, h3, etc.	Seul h1 est stylisé, ce qui peut créer une incohérence visuelle pour les sous-titres.	`h1, article h1, header h1`

c) Controleur (JS)

Défaut	Description	Fichiers concernés
Mauvaise gestion des cookies	L'accès aux cookies (document.cookie) n'est pas sécurisé et pourrait être vulnérable à des attaques XSS.	`function.js, login.js, logout.js, compte.js`
Redirections forcées non optimales	Beaucoup de fichiers redirigent l'utilisateur en vérifiant cookieValue === undefined, mais sans gestion des erreurs ni messages clairs.	`changerMDP.js, compte.js, favori.js, historique.js, panier.js`
Absence de gestion d'erreur dans les requêtes fetch	La plupart des requêtes API (fetch()) ne gèrent pas correctement les erreurs en cas d'échec de requête.	`historique.js, favori.js, produit.js, panier.js, user.js`

Utilisation de innerHTML sans précaution	L'injection directe de contenu via innerHTML peut exposer l'application à des attaques XSS.	`accueil.js`, `produit.js`, `regionApi.js`, `produits.js`
Code dupliqué	Des fonctions similaires (gestion des erreurs, vérification des cookies) sont réécrites plusieurs fois au lieu d'être centralisées.	`compte.js`, `register.js`, `login.js`, `function.js`
Absence de validation côté client	Les formulaires (register.js, changerMDP.js) ne vérifient pas suffisamment les champs avant d'envoyer les données au serveur.	`register.js`, `changerMDP.js`, `ajout_article.js`
Imports et dépendances mal gérés	Certains fichiers importent des modules sans vérification (import { cookieValue } from "./function.js";), ce qui peut causer des erreurs si le fichier function.js est absent.	`accueil.js`, `favori.js`, `panier.js`, `historique.js`
Manque de séparation des responsabilités	Certains fichiers contiennent à la fois de la logique métier et de l'affichage, ce qui rend le code difficile à maintenir.	`accueil.js`, `panier.js, favori.js`

3. Contenu de la partie Client

a) API (PHP)

Défaut	Description	Fichiers concernés
Absence de validation des données	Aucune validation n'est effectuée sur les données envoyées via $_POST, ce qui peut entraîner des problèmes si des données invalides ou manquantes sont envoyées.	`setUser.php`, `setProduit.php`, `setTailProduit.php`, `setColProduit.php`, `newUser.php`
Validation insuffisante de l'email	Le champ email (mel) n'est pas validé correctement pour vérifier qu'il suit le format attendu d'une adresse email.	`setUser.php`, `newUser.php`
Validation insuffisante de la date	La date de naissance (date_naiss) n'est pas validée de manière approfondie, ce qui pourrait permettre d'insérer des dates invalides ou dans un format incorrect.	`setUser.php`, `newUser.php`
Sécurité insuffisante sur les données	Les données sensibles comme les mots de passe ne sont pas vérifiées pour s'assurer qu'elles sont correctement cryptées ou validées.	`newUser.php`
Absence de vérification de l'existence d'un utilisateur	Avant de mettre à jour ou d'insérer des données d'utilisateur, il n'y a pas de vérification si l'utilisateur existe déjà dans la base de données.	`setUser.php`, `setProduit.php`, `setTailProduit.php`, `setColProduit.php`
Manque de gestion des erreurs améliorée	Les messages d'erreur retournés sont parfois trop généraux ou	`payer.php`, `newProduit.php`,

	manquent de détails pour aider à résoudre le problème.	`setProduit.php`, `setUser.php`
Requêtes SQL vulnérables	Certaines requêtes SQL peuvent être sujettes à des attaques par injection SQL si les données utilisateurs ne sont pas correctement échappées ou validées.	`setUser.php`, `newUser.php`
Mise à jour sans vérifier l'impact	Dans certains fichiers, une mise à jour est effectuée sans vérifier si elle a bien modifié des lignes (risque d'échec silencieux).	`setUser.php`, `setProduit.php`, `setTailProduit.php`
Répétition de code	Certaines actions (comme la préparation de requêtes ou l'exécution de la requête avec gestion des erreurs) sont répétées dans plusieurs fichiers, ce qui augmente la maintenance.	`Tous les fichiers`
Manque de confirmation de la commande	Après la commande, il est important de vérifier et confirmer si toutes les étapes ont été correctement réalisées, mais ce n'est pas toujours fait.	`payer.php`
Absence de gestion des erreurs pour certaines requêtes	Parfois, une requête peut échouer sans que l'erreur soit correctement capturée, ce qui peut entraîner des problèmes non signalés.	`payer.php`, `setProduit.php`

b) BDD (SQL)

Défaut	Description	Tables / Triggers / Vues Concernés
Utilisation de MyISAM au lieu d'InnoDB	MyISAM ne supporte pas les transactions et les clés étrangères	`Toutes les tables`
Absence de FOREIGN KEYS	Aucune contrainte d'intégrité référentielle, risque d'incohérences.	`PRODUIT, PANIER, DETAIL_COM, USER`
Stockage des mots de passe dangereux	mdp stocké en claire avec crypt(), salt visible	`USER, SELECT_USERS`
SELECT_USERS expose les mots de passe	Vue accessible affichant mdp et salt	`SELECT_USERS`
AUTO_INCREMENT mal géré	Valeurs AUTO_INCREMENT incorrectes, risque de conflits d'ID	`CATEGORIE, COMMANDE, PRODUIT, USER, TAILLE`
prix_total stocké en DOUBLE	Problème d'arrondi lors des calculs de prix.	`DETAIL_COM, PANIER, SELECT_COMMANDES, SELECT_PANIERS`
Pas de gestion du stock dans TAIL_PROD	Impossible de savoir si une taille est disponible	`TAIL_PROD`
Index absents sur les clés étrangères	Ralentissement des requêtes de jointure.	`PRODUIT, USER, COMMANDE, PANIER`
PANIER ne vérifie pas le stock	Ajout possible même si l'article est en rupture.	`PANIER, PANIER_BEFORE_INSERT`

Jointure inefficace dans SELECT_PANIERS	Requête trop lourde à cause d'une sous-requête inutile.	SELECT_PANIERS
Catégories de tailles mal définies (id_cat = 0)	Certaines tailles ne sont pas correctement liées à une catégorie.	TAILLE
Pas de vérification de l'unicité des emails	Possibilité d'avoir plusieurs comptes avec le même email.	USER
Manque de date_inscription dans USER	Impossible de connaître la date d'inscription des utilisateurs.	USER

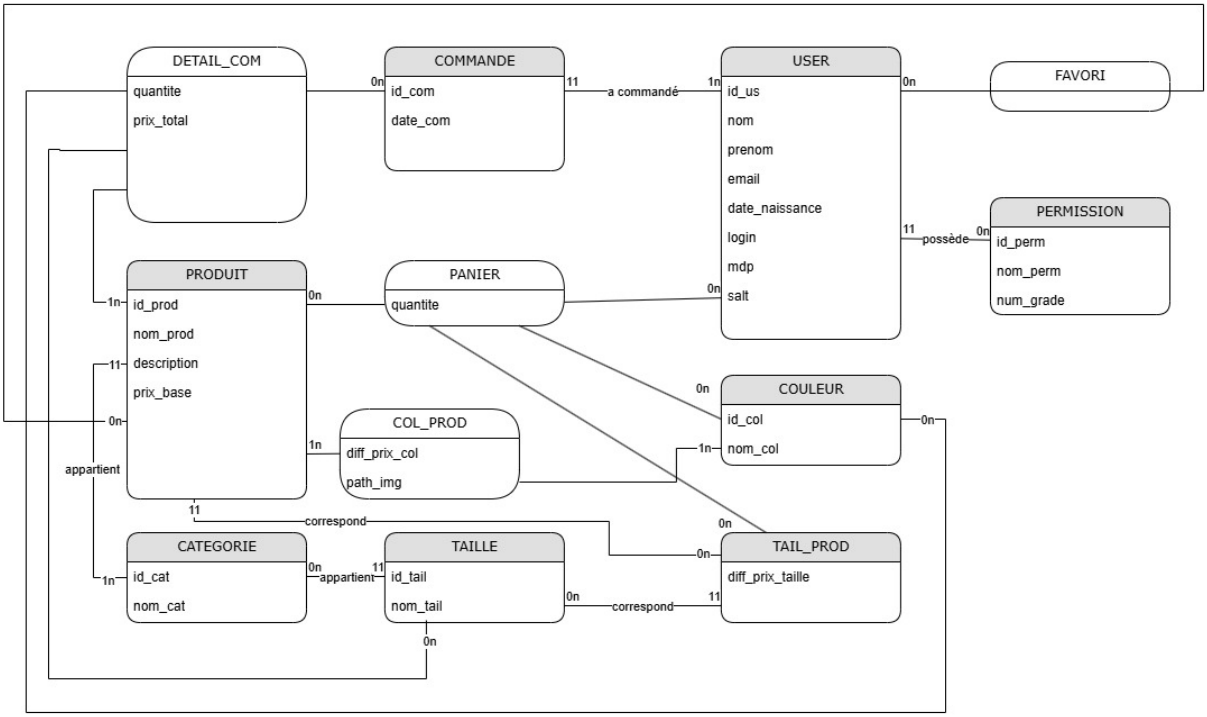
B. Analyse fonctionnelle

Fonctionnalité	État	Détails
Création de compte client	✅	L'inscription fonctionne avec validation.
Accès à la liste des produits	✅	Les produits sont bien affichés avec leurs photos, descriptions et prix.
Recherche par catégories et noms de produits	✅	Le filtrage et le tri avancé fonctionnent bien.
Accès à la fiche détail du produit	✅	On retrouve le SKU, photo, description, tailles, couleurs, prix et stock.
Ajout au panier et aux favoris	✅	Possible via la fiche produit, nécessite d'être connecté.
Accès à la liste de ses favoris	✅	Page dédiée aux favoris enregistrés.
Visualisation des commandes effectuées	✅	L'historique des achats est bien affiché.
Accès aux produits sans connexion	✅	On peut parcourir les produits sans être connecté.
Gestion des variantes (tailles, couleurs)	✅	Lorsqu'un produit a plusieurs tailles ou couleurs, elles apparaissent dans la fiche détail,fait varier le prix.
Connexion/déconnexion avec icône visible	🟠	La connexion et déconnexion fonctionne bien mais L'icône de n'apparaît pas dans l'interface.
Modification du panier avec frais de port	❌	Les frais de port ne sont pas calculés en fonction de la catégorie et du nombre de produits.
Paiement via PayPal sandbox	❌	Aucun paiement simulé n'est géré.
Préparation pour les soldes	❌	Le système ne gère pas les prix barrés ou remisés, ce qui sera utile pour des soldes futures.

C. Planning

Le planning est disponible sur Trello via le lien suivant :
https://trello.com/b/4BAPDac2/sae-401

D. Schéma Entité-Association



E. Schéma relationnel

CATEGORIE (id_cat, nom_cat)
COL_PROD (#id_prod, #id_col, diff_prix_col)
COMMANDE (id_com, date_com, id_us)
COULEUR (id_col, nom_col)
DETAIL_COM (#id_com, #id_prod, #id_col, #id_tail, qte_com, prix_total)
FAVORI (#id_us, #id_prod)
PANIER (#id_us, #id_prod, #id_col, #id_tail, qte_pan)
PERMISSION (id_perm, nom_perm, num_grade)
PRODUIT (id_prod, nom_prod, description, prix_base, id_cat)
TAILLE (id_tail, nom_tail, id_cat)
TAIL_PROD (id_prod, #id_tail, diff_prix_tail)
USER (id_us, nom_us, prenom_us, mel, date_naiss, login, mdp, salt, #id_perm)

Ce schéma définit les relations entre les différentes entités de l'application de vente en ligne de pulls moches de Noël.

Voici une explication détaillée des tables et de leurs interactions :

1. Gestion des Produits

- **PRODUIT (id_prod, nom_prod, description, prix_base, id_cat)**
  - Contient les produits avec leur prix de base.
  - Associe chaque produit à une catégorie via id_cat.
- **CATEGORIE (id_cat, nom_cat)**
  - Définit les catégories de produits (pulls, gants, bonnets).
- **TAILLE (id_tail, nom_tail, id_cat)**
  - Spécifie les tailles disponibles pour certains produits en fonction de la catégorie.
- **TAIL_PROD (#id_prod, #id_tail, diff_prix_tail)**
  - Relation entre un produit et ses tailles possibles, avec un éventuel différentiel de prix.
- **COULEUR (id_col, nom_col)**
  - Liste des couleurs disponibles pour les produits.
- **COL_PROD (#id_prod, #id_col, diff_prix_col)**
  - Relation entre un produit et ses couleurs disponibles, avec une éventuelle variation de prix.

2. Gestion des Utilisateurs

- **USER (id_us, nom_us, prenom_us, mel, date_naiss, login, mdp, salt, #id_perm)**
  - Contient les informations des utilisateurs et leur permission (id_perm).
- **PERMISSION (id_perm, nom_perm, num_grade)**
  - Définit les rôles des utilisateurs (ex. client, administrateur).

3. Gestion des Commandes et du Panier

- **PANIER (#id_us, #id_prod, #id_col, #id_tail, qte_pan)**
  - Associe un utilisateur à un ou plusieurs produits qu'il a ajoutés à son panier, avec gestion des variantes (couleur, taille).
- **COMMANDE (id_com, date_com, id_us)**
  - Contient les commandes passées par les utilisateurs.
- **DETAIL_COM (#id_com, #id_prod, #id_col, #id_tail, qte_com, prix_total)**
  - Détail des commandes avec les produits, quantités et prix total par ligne.

4. Gestion des Favoris

- **FAVORI (#id_us, #id_prod)**
    - Permet aux utilisateurs d'ajouter des produits à une liste de favoris.

Fonctionnalité	Présence dans le schéma	Détails
Connexion/déconnexion avec icône visible	✅USER	Gérée via la table USER avec les champs login, mdp et salt.
Création de compte client	✅USER	L'utilisateur est enregistré dans USER.
Liste des produits avec filtres (catégorie, taille, couleur)	✅PRODUIT, CATEGORIE, TAIL_PROD, COL_PROD	La structure permet bien les filtres demandés.
Affichage des variantes de produits	✅COL_PROD, TAIL_PROD	Géré avec les tables de relation COL_PROD et TAIL_PROD.
Accès à la fiche produit	✅PRODUIT, COL_PROD, TAIL_PROD	Possible en utilisant id_prod pour récupérer les variantes.
Ajout au panier et modification	✅PANIER	Géré par PANIER, avec gestion de la quantité et des variantes.
Visualisation et gestion des commandes	✅COMMANDE, DETAIL_COM	Les commandes et leurs détails sont bien structurés.
Paiement simulé via PayPal Sandbox	❌Non représenté explicitement	Il manque une table pour gérer les transactions simulées.
Gestion des favoris	✅FAVORI	Présent sous forme d'une relation USER ↔ PRODUIT.
Gestion des soldes	❌Non représenté explicitement	Les champs nécessaires (réduction, dates) ne sont pas prévus.

Le schéma relationnel est bien structuré et répond aux besoins essentiels du cahier des charges. Cependant, il manque certains éléments comme la gestion des paiements, des soldes et du stock, qui devront être ajoutés pour être totalement conforme.