



Bouzazi Firas
Oun Mohamed
Zahdi Mohcine
Yanis Bentoumi
Othmane Mounouar
Néo Schobert

Encadré par : [Issam REBAÏ](#)

Wishlist Web App Project

Le projet consiste à développer un site web permettant d'offrir des cadeaux à une personne à partir de sa liste de souhaits.



IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

LA RÉPARTITION DES TÂCHES ENTRE LES DIFFÉRENTS BINÔMES

L'ÉQUIPE EST DIVISÉE EN TROIS BINÔMES :

BOUZAZI FIRAS ET OUN MOHAMED :

- Développement d'une page et de l'entité **Item** (Ajout/Édition/Suppression d'un article dans la wishlist).
- Implémentation de l'entité **PurchaseProof** (Page pour téléverser une preuve d'achat d'un article).

ZAHDI MOHCINE & OTHMANE MOUNOUAR :

- Travail sur l'entité **Wishlist**.
- Création de la page listant toutes les wishlists.
- Création des pages permettant d'éditer et de créer une wishlists (avec la possibilité de supprimer)
- Création de la logique des invitations pour inviter les gens à collaborer sur des wishlists
- Création de logique pour générer des URL pour la page "Gifts purchase"

NÉO SCHOBERT & YANIS BENTOUMI:

- Développement de la **page d'accueil**.
- Mise en place du **système d'administration** (gestion des utilisateurs et tableau de bord des achats).

LE CHOIX TECHNOLOGIQUE DE LA SOLUTION:

Pour la réalisation du projet, nous avons choisi les technologies suivantes :

1. Backend : PHP avec Symfony

○ Pourquoi Symfony ?

- **Framework robuste et structuré** : Permet une architecture MVC claire et facilite l'organisation du code.
- **ORM Doctrine** : Gestion efficace de la base de données avec des entités et des relations bien définies.
- **Sécurité intégrée** : Gestion native des sessions, authentification et protection contre les attaques (CSRF, XSS, SQL Injection).
- **Flexibilité et évolutivité** : Facilité d'ajout de nouvelles fonctionnalités et intégration d'API.
- **Utilisation du générateur de code Symfony Maker** :
 - Génération rapide des entités, contrôleurs et formulaires.
 - Gain de temps et respect des conventions du framework.

2. Frontend : HTML, CSS, Twig

○ Pourquoi Twig ?

- **Système de templating performant** : Séparation claire entre logique et présentation, favorisant la réutilisation des composants.
- **Sécurisé** : Protège contre les injections XSS en échappant automatiquement les variables.

○ CSS pour le design

- Difficultés rencontrées liées à la syntaxe, mais essentiel pour respecter la charte graphique imposée.

3. Base de données : MySQL avec Doctrine ORM

- Stockage structuré des wishlists, items et preuves d'achat.
- Requêtes optimisées grâce à l'ORM pour éviter les injections SQL et améliorer les performances.

4. Hébergement et déploiement

- Utilisation de **Git pour le versioning** et **GitLab/GitHub** pour la collaboration.
- Déploiement sur un serveur **Apache avec PHP sur une machine EC2 du Cloud AWS**

LES POINTS FORTS ET LES POINTS FAIBLES DU PROJET:

Points forts :

1. Fonctionnalités principales implémentées :

- Affichage, création et édition des wishlists opérationnels.
- Création d'items et édition fonctionnelles.
- Lien de co-crédation des wishlists fonctionnel sous certaines conditions.

2. Qualité du code :

- Paradigme **POO** adopté.
- Structuration probable en **MVC**.
- Bonnes pratiques partiellement respectées (indentation, noms explicites des variables et fonctions).

3. Sécurité en partie maîtrisée :

- Filtrage correct des entrées pour les injections SQL.
- Protection contre les failles XSS bien mise en place.

4. Expérience utilisateur :

- Interface apparemment intuitive et JavaScript côté client utilisé.
- Présence d'un dashboard administrateur.

Points faibles :

1. Problèmes fonctionnels :

- **Suppression des wishlists et items non fonctionnelle** (boutons inactifs ou mauvaise redirection).

- **Problèmes avec le login/logout** (failles dans la validation des entrées, absence de bouton logout sur toutes les pages).
- **Vérouillage d'un utilisateur inefficace** (un utilisateur bloqué peut toujours se connecter).
- **Affichage des preuves d'achat non fonctionnel.**
- **Filtrage et tri des items non opérationnels.**

2. Problèmes de sécurité :

- **Absence de protection CSRF** sur plusieurs formulaires (login, inscription).
- **Accès direct aux wishlists via URL sans restrictions.**

LES POINTS NON TRAITÉS DANS LE PROJET:

- On a pas réussi à créer la page de gifts purchase dans le temps qui est censé être la page pour acheter les gifts pour les autres.

LES DIFFICULTÉS RENCONTRÉES POUR CHAQUE BINÔME

BOUZAZI FIRAS ET OUN MOHAMED :

Avancement anticipé par rapport au reste du groupe

- En ayant commencé le développement plus tôt, nous avons pu créer les entités **Item** et **PurchaseProof**.
- Cependant, nous n'avons pas pu intégrer ces fonctionnalités dans les autres parties du projet, car les autres membres du groupe n'avaient pas encore avancé.
- Cela a empêché la mise en place de la page permettant d'ajouter un **Item** à une **wishlist**, ainsi que l'ajout d'un **PurchaseProof** à un **Item** (ce qui le supprime de la wishlist).

Difficultés avec le CSS

- Nous avons rencontré des problèmes au niveau du **CSS**, notamment parce que nous avons oublié certaines règles de syntaxe.
- Cela nous a fait perdre du temps pour styliser correctement les pages et respecter la charte graphique imposée.

ZAHDI MOHCINE & OTHMANE MOUNOUAR :

Problèmes de lancement de serveur pour les tests

- Nous avons eu du mal à lancer le serveur pour charger et tester le rendu des pages
- Par conséquent, les tests devaient être effectués sur une machine tierce (celle d'un autre membre du groupe)

Problèmes techniques

- On est allé sur des techniques un peu nouveaux qui demandaient du temps pour les comprendre et déboguer (la création des URLs, l'encodage et le décodage de tokens, manipulation de la clipboard par navigateur)
- Il y avait quelques subtilités à connaître à propos de Doctrine. Par exemple, le requêtage des repository est fait par le nom des attributs non par les noms des columns dans la base de données, ainsi, il fallait comprendre la relation existante entre les attributs et les effets tel que le delete par Cascade. Il faut également être attentif aux différences entre les données dans la mémoire et dans la base de données et savoir les synchroniser.

NÉO SCHOBERT & YANIS BENTOUMI:

Problèmes liés à la base de données

- On a rencontré quelques soucis avec la configuration de Doctrine et la gestion des migrations. Certaines tables ne se créaient pas comme prévu à cause d'erreurs dans les relations entre les entités.
- Du coup, on a dû passer pas mal de temps à corriger la structure de la base et à ajuster les fichiers de migration pour éviter des conflits lors des mises à jour du schéma.