

NOM	VINCENT
Prénom	Francois
Date de naissance	18/12/61

Copie à rendre

Bachelor Développeur 22/24

Android, Angular, Flutter, Front End, Full Stack, IOS, PHP /Symfony

Partie 1 : livrables

1. Entrez le lien de votre outil de gestion de projet

<https://francoisvincent.atlassian.net/jira/software/projects/ECF/boards/1>

2. Entrez le lien de votre git

<https://github.com/u014697/ECF>

Le git contient les éléments suivants :

- l'ensemble des fichiers php nécessaire au bon fonctionnement de l'application.
- Un template de DSN pour renseigner les éléments de connexion à la BDD (fichier ref/connect.php)
- un script pour la création d'utilisateurs de démo, mais qui pourrait être customiser avec de vraies données (initusers.php)
- un fichier .sql pour la création des tables (fichier sql/create.sql)
- un fichier .sql montrant une transaction (fichier sql/exemple transaction.sql). L'explication de cette transaction se trouve dans le dossier technique (lui même dans le git)
- la charte graphique au format pdf (fichier documents/ECF_bachelordev22-24_charte_graphique.pdf)
- le dossier technique (fichier documents/ECF_bachelordev22-24_documentation_technique.pdf)
- le manuel d'utilisation (fichier document/ECF_bachelordev22-24_manuel_utilisation.pdf)

le document de gestion de projet est constitué de la partie 2 de ce document.

3. Entrez le lien de vos applications déployées en ligne

mockup parcours visiteur : <https://xd.adobe.com/view/1373dc33-7a20-4ffa-9a31-f8a6eda538e5-4620/>

mockup parcours utilisateur : <https://xd.adobe.com/view/27c71516-75fc-432d-9b8d-e003a71c9dd0-7294/>

mockup parcours administrateur : <https://xd.adobe.com/view/53f380dc-f17b-4ef5-8216-af5f4b67e34d-06a0/>

mockup parcours employé : <https://xd.adobe.com/view/28cf1a72-6f1b-499a-ab90-a2fa8fb28a17-7d9c/>

site WEB : <https://ecf-fv.go.yj.fr/acceuil.php>

sur ce site , quelques catégories et quelques produits ont été créés, ainsi qu'un administrateur, des employés et des utilisateurs pour faciliter la démo ...

les éléments créés sont les suivants en déroulant la procédure d'installation de l'application (voir le readme.md sur le git.

initialisation des tables de l'application Ventalis.

IMPORTANT, les tables doivent être vides avant de lancer ce script. Mot de Passe des utilisateur = Mot-2-Passe

creation Administrateur bigboss@ventalis.com pass=Mot-2-Passe

creation Employé johnventalis@ventalis.com pass=Mot-2-Passe

creation Employé janeventalis@ventalis.com pass=Mot-2-Passe

creation Employé jerryventalis@ventalis.com pass=Mot-2-Passe

creation Utilisateur johndoe@jd.com pass=Mot-2-Passe

creation Utilisateur janedoe@jd.com pass=Mot-2-Passe

creation Utilisateur jerrydoe@jd.com pass=Mot-2-Passe

creation catégorie 'haut de gamme'

creation catégorie 'entrée de gamme'

creation catégorie 'bon rapport qualité prix'

creation du produit : 'Produit No1'

creation du produit : 'Produit No2'

creation du produit : 'Produit No3'

a noter que le mail n'est pas fonctionnel, (seule la messagerie interne à l'application l'est) il n'y a donc pas d'envoi de mot de passe par mail, ceux ci seront donc connus à l'avance dans la procédure d'inscription utilisateur pour émuler l'envoi d'email: **Nouveau-Mot2Passe**

les fichiers image des produits doivent être au format .png

Partie 2 : Planification

1. Comment avez-vous effectué la planification de votre projet ?

A la lecture du cahier des charges, j'ai créé dans mon Kanban sous Jira les différentes tâches à réaliser, à savoir la réalisation des différentes user stories, l'écriture des différents documents.

Chaque tâche est ensuite déplacée dans la colonne adhoc en fonction de son avancement. Ceci me permet de bien m'assurer de ne rien oublier !

Ma première tâche a été le maquetage sous Adobe XD, ceci m'a permis de m'imprégner du cahier des charges et de bien comprendre ce qui est attendu. Il est d'ailleurs à noter que certaines non-conformités sont présentes dans les mockups, mais j'ai décidé de ne pas les reprendre post-réalisation faute de temps et d'utilité.

Ensuite est venu le temps pour moi de concevoir le modèle de données. La ligne directrice est d'éviter toute duplication des données dans ce SGBDR.

La tâche suivante a été de réaliser le template des différentes pages du front-end en HTML CSS. L'idée étant d'avoir un modèle facilement réutilisable pour chaque nouvelle fonction.

Une fois le développement et les tests réalisés, il m'a fallu trouver une solution d'hébergement (chez Planet Hosters), le mettre en place, et faire les quelques adaptations qui se sont avérées nécessaires.

2. Quelle méthode de gestion de projet avez-vous utilisée et pourquoi ?

Travaillant tout seul sur ce projet, un simple tableau Kanban était largement suffisant. En effet, les buts principaux de cette gestion de projet étaient :

- n'oublier aucun livrable dans le stress
- mesurer régulièrement la taille des tâches encore à réaliser pour gérer mon temps.

La mise en place de Scrum ne me semblait pas apporter de valeur dans ce cadre.

Pour les mêmes raisons, il ne m'a pas semblé utile de passer du temps à faire un Gantt, toutes les tâches étant séquentielles.

Partie 3 : Technologies utilisées

- 1. Quelles technologies avez-vous utilisées ? Précisez tous les éléments qui vous ont permis de faire ce choix. Ajoutez également un paragraphe expliquant en quoi ces choix sont le plus adaptés pour la demande spécifiée dans l'énoncé.**

J'ai choisi d'utiliser HTML/CSS pour le front, il n'y a en effet pas d'autre choix. Je ne prévoyais de mettre en œuvre de framework, car ceux-ci peuvent s'avérer complexes à paramétrer pour obtenir le rendu désiré lors du design, et le budget (temps) est très serré. Au final, j'ai intégré un menu responsive bootstrap. C'est un peu lourd et ça manque de souplesse, mais ça fonctionne bien.

Pour effectuer une application mobile, j'ai fait le choix de rendre le site « responsive ». En effet, ceci permet un substantiel gain de temps en réutilisant les mécanismes d'authentification notamment. L'utilisation de media query permet notamment d'améliorer la responsivité sur des orientations portrait. Pour obtenir un menu responsive j'ai utilisé bootstrap, ce qui m'a quelque peu contraint sur le design de la barre de menu, mais cela m'a fait gagner pas mal de temps.

En outre cette solution permet d'adresser aussi bien les flottes Android qu'iOS !

Le principal handicap de ce choix sera de devoir taper l'URL pour y accéder, ce qui n'est pas pratique sur mobile, mais une icône de raccourci peut être facilement installée en suivant la procédure suivante : <https://www.clubic.com/tutoriels/article-891621-1-comment-ajouter-raccourci-web-page-accueil-smartphone-android.html>

Pour le backend, il est développé en PHP, qui est le langage le plus simple, et qui est directement utilisable sur la plupart des environnements d'hébergement (y compris gratuit, car il me fallait pouvoir mettre le site en ligne)

Une base relationnelle MariaDB permet en complément le stockage, mais l'accès aux données est rigoureusement effectué au moyen de requêtes préparées PDO pour maximiser la portabilité en cas de changement d'environnement SQL.

- 2. Quelles mesures avez-vous prises afin de protéger vos applications des potentielles vulnérabilités de sécurité ? (Injection SQL, failles XSS, ...)**

D'une part, l'utilisation systématique de requêtes préparées PDO va nous garantir contre les injections SQL grâce aux méthodes bind sur les paramètres.

Concernant les failles XSS, j'ai filtré avec htmlspecialchars les valeurs externes lors de leur stockage dans la BDD. Ainsi, lors des affichages de données issues des requêtes de lecture dans le SGBDR, les données sont inoffensives.

Pour plus de sécurité, il aurait fallu que j'installe bootstrap sur mon serveur d'hébergement. Ca serait à faire dans la vraie vie pour ce protéger d'un hack (improbable) des serveurs de référence bootstrap.

Pour la gestion de l'authentification, le choix fait est de développer les mécanismes simples de connexion initiale puis de gestion d'un token de connexion.

La connexion initiale utilisera les fonctions PHP password_hash et password_verify pour stocker/vérifier le hash des mots de passe. Ces fonctions sont suffisamment « lentes » pour éviter une attaque en force brute. A noter que ces fonctions gèrent automatiquement le salage.

A la connexion, un token est généré et stocké en tant que variable de session pour assurer le suivi. Bien sûr, c'est l'empreinte hash en md5 qui est stockée dans la bdd.

Ce token sera utilisé à chaque page pour reconnaître l'utilisateur et son rôle.

Une connexion https sera nécessaire pour assurer la protection de ce token.

Enfin, ce token n'aura une durée de vie que de 2 heures après le dernier accès à l'application, après quoi, la variable de session sera « unset »

De la même façon, j'ai prévu un bouton de déconnexion (même si ça n'était pas demandé, mais s'était indispensable pour les tests et changer d'utilisateur!). A la déconnexion, la variable de session est supprimée, et la date de péremption est forcée dans la bdd.

En supplément, les menus destinés aux autres rôles d'utilisateurs ne sont pas présentés (même si la protection apportée est assez symbolique), mais Le rôle de l'utilisateur est systématiquement vérifié sur toutes les pages à protéger (en cas de saisie directe de l'url)

last but not least, j'ai mis un fichier index.php dans les répertoires contenant du code afin d'éviter qu'un petit malin ne puisse tout simplement accéder aux fichiers sources.

La politique de mdp « forts » demandée est appliquée, aussi bien dans le formulaire

```
<input type="password" id="password" name="password" required pattern="(?!.*?[A-Z])(?!.*?[a-z])(?!.*?[0-9])(?!.*?[#?!@$%^&*~]).{8,}" />
```

que coté backend :

```
if (!preg_match('/^(?!.*?[A-Z])(?!.*?[a-z])(?!.*?[0-9])(?!.*?[#?!@$%^&*~]).{8,}$/', $pass)) return false;
```

3. Décrivez les tests applicatifs que vous effectuez et dans quel but.

Au niveau des tests, je commence par les tests unitaires des fonctions d'accès aux données. Il faut pour cela construire un jeu de données de test qui va permettre de tester toutes les branches de la fonction en question.

Prenons l'exemple de la fonction suivante qui est destinée à retourner les propriétés de l'utilisateur connecté :

```
function getUser () {  
    global $pdo;
```

```
try {
    if (isset ($_SESSION["token"])) {
        $getuser = $pdo->prepare("select * FROM users WHERE hashToken = :hashtoken");
        $getuser->bindValue(':hashtoken', hash('md5', $_SESSION["token"]), PDO::PARAM_STR);
        $getuser->execute();
        $user= $getuser->fetch(PDO::FETCH_ASSOC);
        return $user;
    }
    return false;
}
catch (PDOException $e) {
    return false;
}
}
```

Le jeu de test unitaire va consister en les tests suivants :

pas d'utilisateur connecté (la variable de session n'existe pas) → la fonction retourne false
le token n'est pas dans la base (pour cela, avant d'appeler la fonction dans le script de test, on altère la variable de session → la fonction retourne false.

On génère une erreur PDO en appelant la fonction sans s'être connecté à la base → la fonction retourne false.

Enfin, avec un token non trafiqué d'un utilisateur connecté et la base ouverte → on récupère un tableau associatif contenant les propriétés de l'utilisateur connecté.

Les tests de rendu du front end sont réalisés pour chaque « page » en modifiant la taille d'écran et l'orientation. On vérifie alors visuellement qu'il n'y a pas d'anomalie pour différents « flagship » du marché (PC, Samsung S20 et iPhone notamment).

Enfin, on déroule les différentes user stories du cahier des charges pour s'assurer comportement attendu, en vérifiant à chaque étape la mise à jour de la BDD.

J'ai pris soin également de tester les conditions aux limites, et notamment le cas où le résultat des requetes est vide (par exemple, aucun message reçu ou panier vide, ...)

4. Comment avez-vous effectué le déploiement de votre application ?

Pour déployer mon site web, j'ai recherché un hébergeur. Mes critères de recherche étaient les suivants :

- hébergement gratuit
- PHP disponible
- base de donnée SQL
- PhpMyAdmin
- pas de publicité insérée dans les pages

mon choix s'est porté sur l'offre gratuite de planet hoster <https://www.planethoster.com/fr/World-Lite>





évidemment pour ce prix, il ne faut pas être difficile sur le nom de domaine « ecf-fv.go.yj.frv » mais à part ça, ça fait ce dont j'ai besoin.

Sur le site d'exploitation de l'hébergeur, j'ai créé une base de donnée et un utilisateur qui me serviront le login pour l'accès à cette base.

BASES DE DONNÉES ACTUELLES

Créer

Rechercher

BASES DE DONNÉES	TAILLE	UTILISATEURS AVEC PRIVILÈGES	ACTIONS
mwryvfpk_ventalis	12 kB	mwryvfpk_superuser  	 

Ensuite, j'ai utilisé **filezilla** avec les identifiants ftp fournis par l'hébergeur pour copier les différents fichiers et répertoires du site dans le répertoire distant public_html.

Une fois ceci réalisé, j'ai pu créer les tables à l'aide du script sql utilisé pour les créer sur l'environnement de développement (fichier sql/create.sql dans le git). Pour cela, je le copie/colle dans l'onglet « sql » de phpmyadmin puis je clique sur « executer ».

Il ne reste plus qu'à créer jeu minimum de données de démo en lançant le script initusers.php.

initialisation des tables de l'application Ventalis.

IMPORTANT, les tables doivent être vide avant de lancer ce script. Mot de Passe des utilisateur = Mot-2-Passe

creation Administrateur bigboss@ventalis.com pass=Mot-2-Passe

creation Employé johnventalis@ventalis.com pass=Mot-2-Passe

creation Employé janeventalis@ventalis.com pass=Mot-2-Passe

creation Employé jerryventalis@ventalis.com pass=Mot-2-Passe

creation Utilisateur johndoe@jd.com pass=Mot-2-Passe

creation Utilisateur janedoe@jd.com pass=Mot-2-Passe

creation Utilisateur jerrydoe@jd.com pass=Mot-2-Passe

creation catégorie 'haut de gamme'

creation catégorie 'entrée de gamme'

creation catégorie 'bon rapport qualité prix'

creation du produit : 'Produit No1'

creation du produit : 'Produit No2'

creation du produit : 'Produit No3'

il ne me restait plus qu'à supprimer ce script du serveur pour éviter que quelqu'un d'autre ne se crée un compte administrateur ! Ce script est disponible sur le git.

Lors du déploiement j'ai quand même rencontré 2 problèmes.

- 1) sur XAMPP, les noms de tables supportent les majuscules, mais pas sur l'environnement de prod, ce qui mettait les requêtes en échec, j'ai donc du tout remettre en minuscules, heureusement que toutes les requêtes sont dans un même fichier
- 2) un problème de gestion des caractères accentués se produisait sur les données lues depuis la BDD de production. Google m'a assez vite sorti de cet embarras et j'ai réglé le problème en ajoutant ; charset=utf8mb4 dans le DSN de production.

Le site est accessible avec l'url suivante :

<https://ecf-fv.go.yj.fr/acceuil.php>