PROJET 8
Plateforme Nutella

Jean baptiste Servais

Nous allons vous décrire notre projet comme si nous étions un utilisateur qui rencontre le site pour la première fois. L'utilisateur peut : s'inscrire, rechercher des aliments, visualiser des aliments, voir la description de l'aliment sur Openfactfood et remplacer un de ses aliments.

Nous devons donc nous inscrire. Pour cela nous sommes dans le dossier accounts. Plus précisément dans la fonction register_view(). Dans cette fonction nous importons depuis forms.py les formulaires d'inscription UserRegisterForm().

Ici nous définissons les input à remplir depuis le model User de base en appelant la classe Meta et nous avons choisis username, email, email et password comme champs d'inscription. Nous nettoyons les données et affichons des erreurs (email qui existe déjà, email non identique). Revenons à view.py. Si on recoit une requete POST et que le formulaire est valide alors on récupére l'username, on nettoie le password, on redéfinit le password de celui généré par django à celui généré par l'utilisateur et on sauvegarde. Ensuite on récupère l'username que nous enregistrons dans foodAccount que nous avons défini dans une nouvelle table foodAccount. Pour cela nous avons créer un modèle depuis models.py. Ici nous avons définis un username avec 6 aliments. Nous identifions directement l'user que nous retournons vers la page d'acceuil. Pour résumer UserRegisterForm(), nous avons mis 4 inputs sur notre page html grâce au modèle de base de django. Si le formulaire est valide on sauvegarde notre utilisateur pour la connexion et on le sauvegarde pour ses aliments (A noter que nous aurions pu faire un one-to-field avec le modèle de base). Enfin nous l'identifions et le redirigeons vers la page d'accueil).

Imaginons la deuxième visite de l'utilisateur et qu'il se soit déconnecté en appuyant sur le logo de déconnexion qui fera appel à logout_view(). L'utilisateur doit se connecter. Il appuie sur le logo de connexion et est dirigé vers login.html. Il doit saisir deux inputs l'username et le password. Les informations sont envoyé a login_view() qui va voir si le formulaire est valide et s'il l'est on redirige l'utilisateur vers la page d'acceuil sinon on reste sur cette page.

Pour résumé 2 models sont appelés le modele de base pour les user et la connexion et le modele pour les aliments de l'utilisateur. 2 formulaires l'un pour l'enregistrement et l'autre pour la connexion.

Une fois connecté l'utilisateur doit pouvoir chercher des aliments. Depuis la page recherche.html l'utilisateur rentre un aliment soit par la barre de recherche en haut soit sur la page d'accueil. Cela envoie les informations à la fonction searching() du fichier views.py via une requete de type POST par l'appuie sur entrée d'un input de type submit. On récupère la recherche ainsi que le nom de l'utilisateur par une balise caché en html ainsi que sa recherche.

La fonction searching() propose deux choses l'une est d'afficher la recherche de l'aliment ainsi que les 6autres possibilités d'aliment de nutriscore le plus élevé de la catégorie. L'autre de stocker un des aliments de recherche dans les aliments de l'utilisateur.

Pour la première étape, on doit vérifier les aliments de l'utilisateur, c'est à dire qu'on vérifie qu'il n'a pas plus de 6 aliments. S'il y a plus de 6 aliments alors on affiche un message à l'utilisateur. Ainsi s'il essaie de rentrer l'aliment il y aura un retour négatif grâce à un fonction js. Ensuite on récupère l'image de l'aliment (pour le fond d'écran) recherché ainsi que son nom (pour le titre ex : ferrero rocher). Ensuite on fait appel à la fonction better_nutri() qui permet de récupérer les informations de la recherche, de filtrer nos aliments via la catégorie de la recherche, on récupère les 20 premiers trié par nutriscore. On affiche la recherche plus cinq autres produits. (Pourquoi 20 ? au cas ou il n'y aurait pas d'image ou de nutriscore ect...). Ensuite nous affichons les données : l'image, le nutriscore et le nom dans les balises prévus sur la page html.

La deuxième option est de mettre les aliments recherchés dans les aliments de l'utilisateur Pour cela même système sauf que nous vérifions que l'utilisateur n'a pas plus de 6 items et qu'il ne possède pas déjà deux fois

le même produit par verification_product_no_two() du fichier algo_open.py. On vérifie les aliments de l'utilisateur et si le produit est égale à un de ses aliments alors on renvoie False (il l'a déjà) sinon True (il peut l'enregistrer).

L'utilisateur peut enfin accéder à ces produits. Il ne pouvait pas avant (on lui disait d'enrichir son panier de 6 éléments). On récupère les informations de ses aliments et on les affiche même procédé. Maintenant il veut les remplacer. Il appuie sur l'image d'un des produit, on lui affiche 3 options : « non rien » on ré affiche l'image, » voir les détails » on l'envoie sur Openfactfood grâce au titre et au code récupérer depuis la database. Sinon il veut le « remplacer le produit ». Depuis la page html on récupère le produit que l'utilisateur veut remplacer. On récupère l'utilisateur connecté. Et on vérifie que le produit n'est pas déjà présent. Si non, on remplace le produit par l'ancien produit.

Pour rappel l'utilisateur a cliqué sur un de ces produit. Il a cliqué sur remplacer qui affiche les 6 produits avec un nutriscore le plus élevé de la catégorie l'utilisateur appuie sur remplacer et on remplace le nouveau produit par l'ancien en database.

Les difficultés qui ont été rencontrés sont : l'apprentissage de Django, l'orm, les différentes techniques liées à Django (connexion, déconnexion). Se servir des base de donnée postgresql.

Les améliorations : le design, plus de fonctionnalité, un tchat, des tutoriels cuisine...

Heroku : https://openfactfood.herokuapp.com/
Github : https://github.com/pastrouveedespeudo/projet-8