Adrian Duant

Formation Coding

Dossier Projet

Développeur web et web mobile

**Sommaire :**

* Liste des compétences couvertes par le projet (p.3)
* Résumé du projet (p.4)
* Cahier des charges (p.5)
* Spécifications fonctionnelles du projet (p.8)
* Spécifications techniques du projet (p.11)
* Extraits de code significatifs (p.14)
* Veille sur les failles de sécurité (P.19)
* Description d’une situation de travail ayant nécessité une recherche sur un site anglophone (p.20)
* Traduction d’un extrait du site anglophone utilisé pour effectuer la recherche précédente (p.21)

**Liste des compétences couvertes par le projet :**

Développer la partie front-end d’une application web ou web mobile en intégrant les recommandations de sécurité :

* Maquetter une application
* Réaliser une interface utilisateur web statique et adaptable
* Développer une interface utilisateur web dynamique

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

**Résumé du projet :**

Le projet est la partie forum d’un site web.

Les utilisateurs peuvent accéder au forum seulement s’ils sont connectés au site web.

Pour accéder aux discussions, les utilisateurs passent d’abord par une page contenant toutes les sections du forum, puis sur une page contenant toutes les discussions de la section sélectionnée d’où ils peuvent accéder à la discussion souhaitée.

Les discussions sont triées en fonction de la date et de l’heure de la dernière réponse ayant été postée dessus.

La date et l’heure de ce dernier message sont affichées à côté du titre de la discussion sur la page regroupant toutes les discussions d’une section.

Les utilisateurs sont répartis en trois rôles : administrateur, modérateur et membre.

Les membres peuvent :

* Voir les discussions
* Participer aux discussions
* Créer des discussions

Les modérateurs bénéficient des droits des membres et peuvent en plus :

* Supprimer des messages
* Supprimer des discussions
* Bannir des utilisateurs
* Amnistier les utilisateurs bannis

Les administrateurs bénéficient des droits des modérateurs et peuvent en plus :

* Attribuer les rôles aux utilisateurs

Les bannissements et les rôles sont gérés sur un onglet différent du forum.

Sur cette page, les modérateurs ne peuvent voir que la partie gérant les bannissements des utilisateurs.

Les administrateurs peuvent, eux, voir la partie permettant d’attribuer les rôles en plus de la partie gérant les bannissements.

**Cahier des charges :**

Le site web est un site de jeu textuel contenant des évènements aléatoires.

Mon collègue Matthieu Guertener s’occupe de la partie principale du site ainsi que du jeu, et je m’occupe de la partie forum.

Le forum n’est accessible qu’aux utilisateurs connectés.

Il est composé de plusieurs catégories. Chaque catégorie contient une ou plusieurs sous-catégorie. Les sous-catégories contiennent les sujets créés par les utilisateurs.

Les sujets sont créés avec un message auquel les autres utilisateurs peuvent répondre à l’aide de commentaires et ainsi instaurer une discussion.

Les utilisateurs sont répartis en trois rôles : administrateur, modérateur et membre.

Tous les utilisateurs peuvent :

* Voir les discussions
* Participer aux discussions
* Créer des discussions

En plus de ceci, les modérateurs et les administrateurs peuvent :

* Supprimer des messages
* Supprimer des discussions
* Bannir des utilisateurs
* Amnistier les utilisateurs bannis

Et les administrateurs peuvent aussi :

* Attribuer les rôles aux utilisateurs

Les utilisateurs ont soit le statut libre, soit le statut banni. Les utilisateurs ayant le statut banni ne peuvent pas accéder au forum.

Une catégorie est définie par son nom.

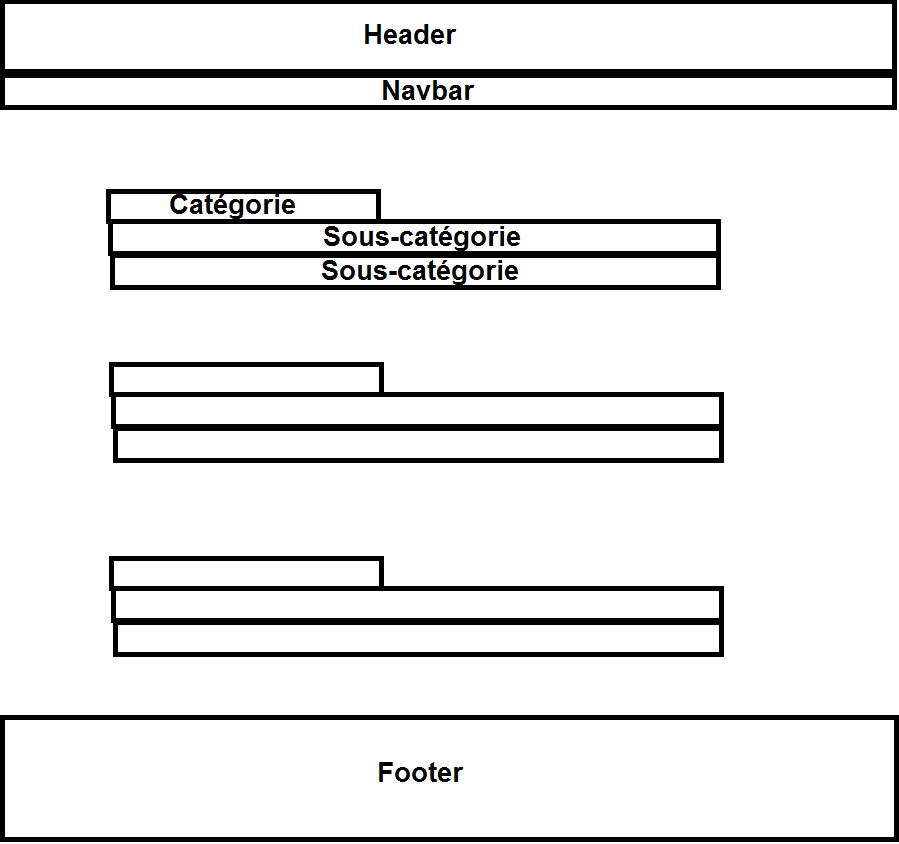
Une sous-catégorie est défini par son nom et la catégorie à laquelle elle appartient.

Un sujet est défini par son titre, son auteur, sa date de création, la sous-catégorie à laquelle il appartient et son message.

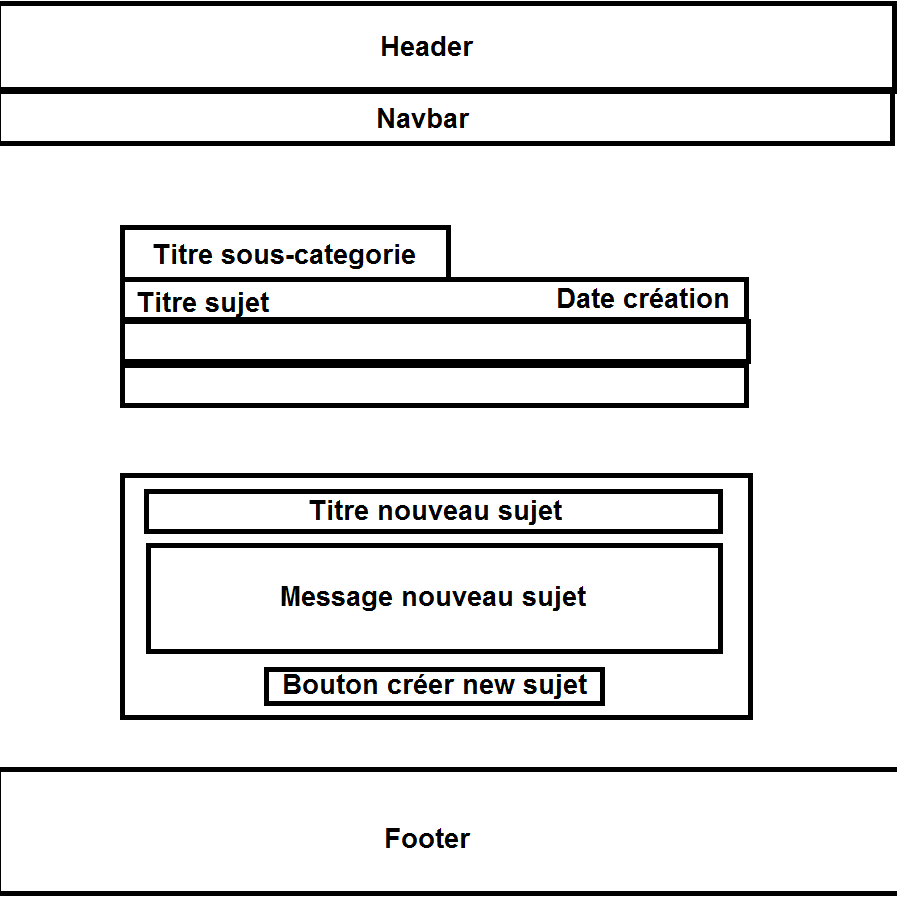
Un commentaire est défini par son texte, son auteur, sa date d’envoi et le sujet auquel il répond.

Un utilisateur est défini par son pseudonyme, son mot de passe, son statut et son rôle.

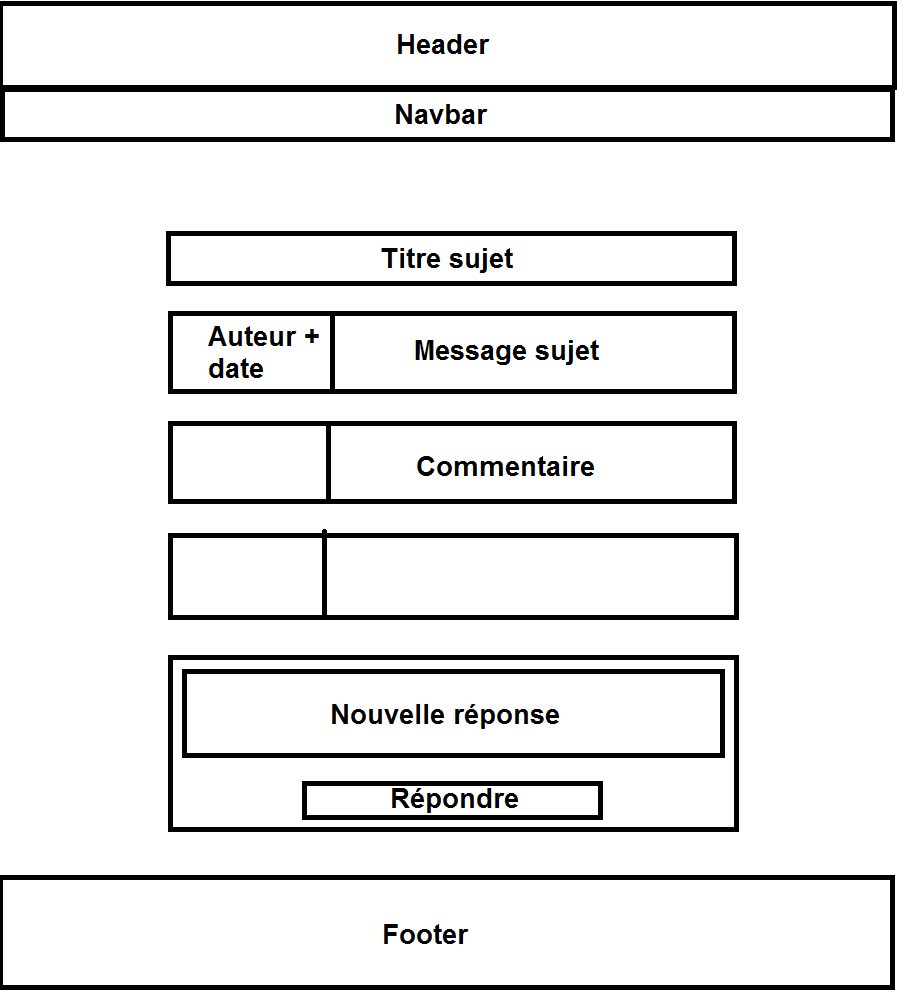
Maquette du forum :



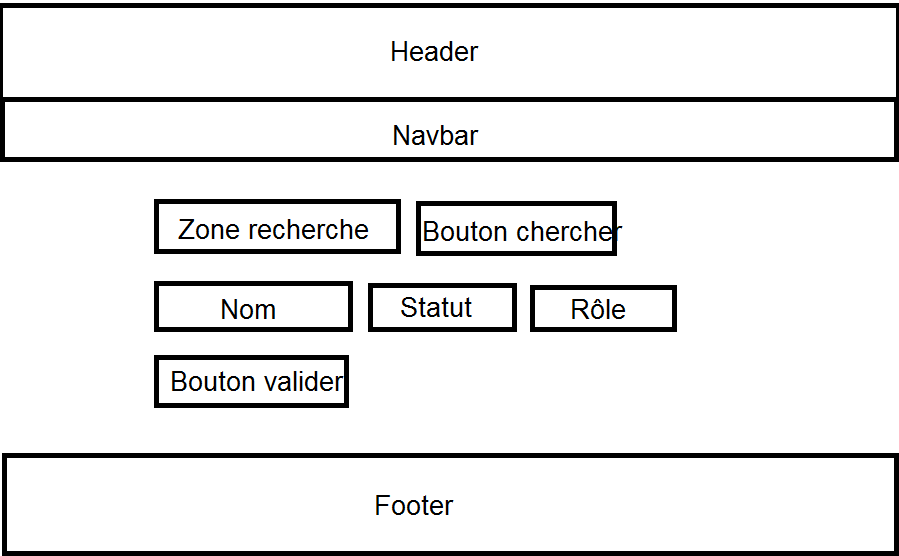
Page : index



Page : sous-catégorie



Page : sujet

****

Page : statut

**Spécifications fonctionnelles du projet :**

Arborescence :

Voici l’arborescence de l’application

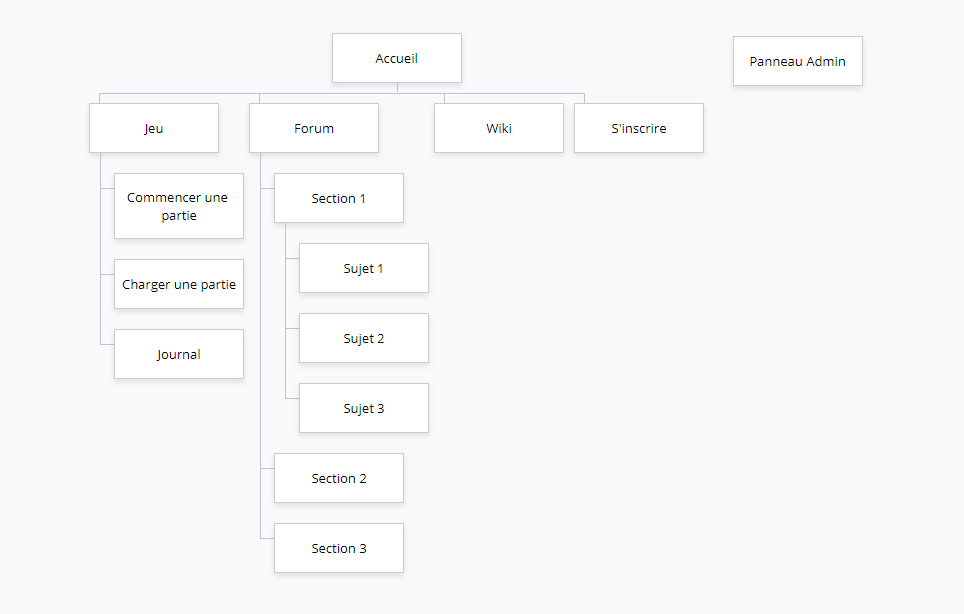
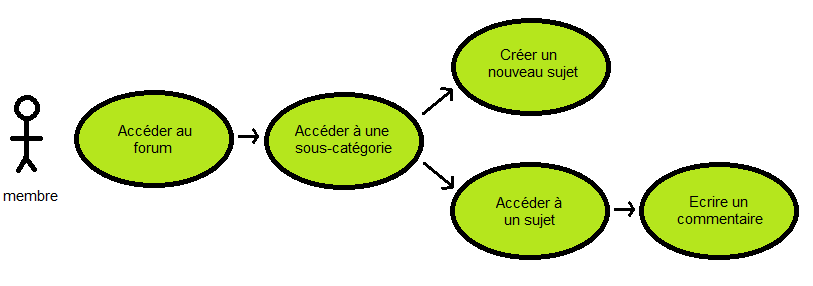
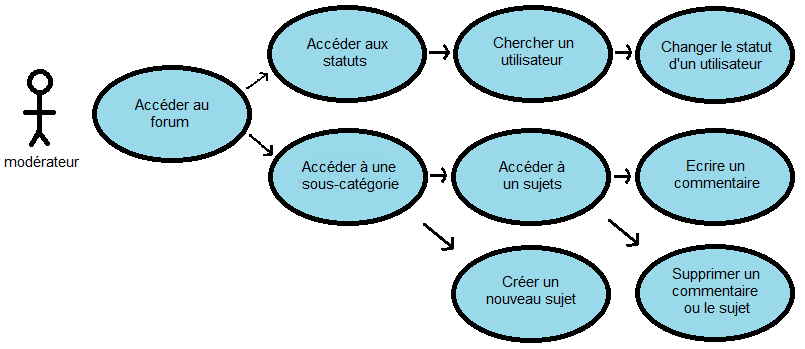
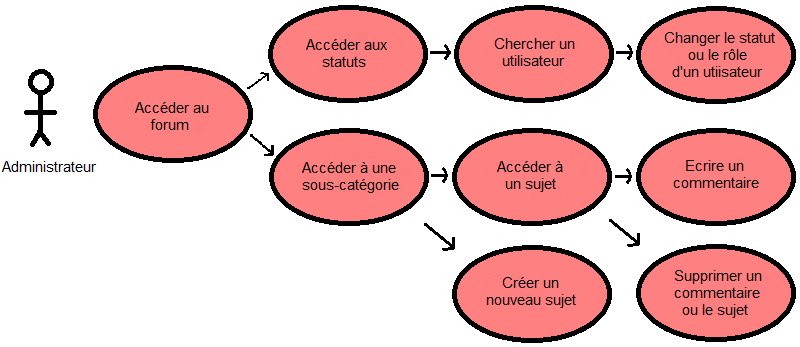


Schéma des cas d’utilisation :

****

****

****

Cas d’utilisation d’un membre :

* Le membre accède au forum depuis le site. Il accède ensuite à la sous-catégorie qui l’intéresse, puis à un sujet de cette sous-catégorie. Il lit les précédents messages de cette conversation puis y répond.
* Le membre accède au forum depuis le site. Il accède à la sous-catégorie qui l’intéresse et crée un nouveau sujet dans celle-ci avec un premier message.

Cas d’utilisation d’un modérateur :

* Le modérateur accède au forum, puis à une sous-catégorie et crée un nouveau sujet.
* Le modérateur accède au forum, puis à la sous-catégorie qui l’intéresse. Il accède ensuite à un sujet, lit la discussion en cours et y répond.
* Le modérateur accède au forum, puis à une sous-catégorie, puis à un sujet. Il voit un commentaire inadapté et le supprime. Si le sujet est inadapté, il supprime le premier message du sujet.
* Le modérateur accède au forum, puis à la page de gestion des statuts. Il cherche le membre dont il souhaite changer le statut grâce à son nom et change son statut.

Cas d’utilisation d’un administrateur :

* L’administrateur accède au forum, puis à une sous-catégorie et crée un nouveau sujet.
* L’administrateur accède au forum, puis à la sous-catégorie qui l’intéresse. Il accède ensuite à un sujet, lit la discussion en cours et y répond.
* L’administrateur accède au forum, puis à une sous-catégorie, puis à un sujet. Il voit un commentaire inadapté et le supprime. Si le sujet est inadapté, il supprime le premier message du sujet.
* L’administrateur accède au forum, puis à la page de gestion des statuts. Il cherche le membre dont il souhaite changer le statut ou/et le rôle grâce à son nom et change son statut ou/et son rôle.

**Spécifications techniques du projet :**

Choix technologiques :

Pour réaliser ce projet, j’ai utilisé un ordinateur sous Windows 7 et un autre sous Windows 10.

Pour la partie front-end du projet, j’ai utilisé les langages :

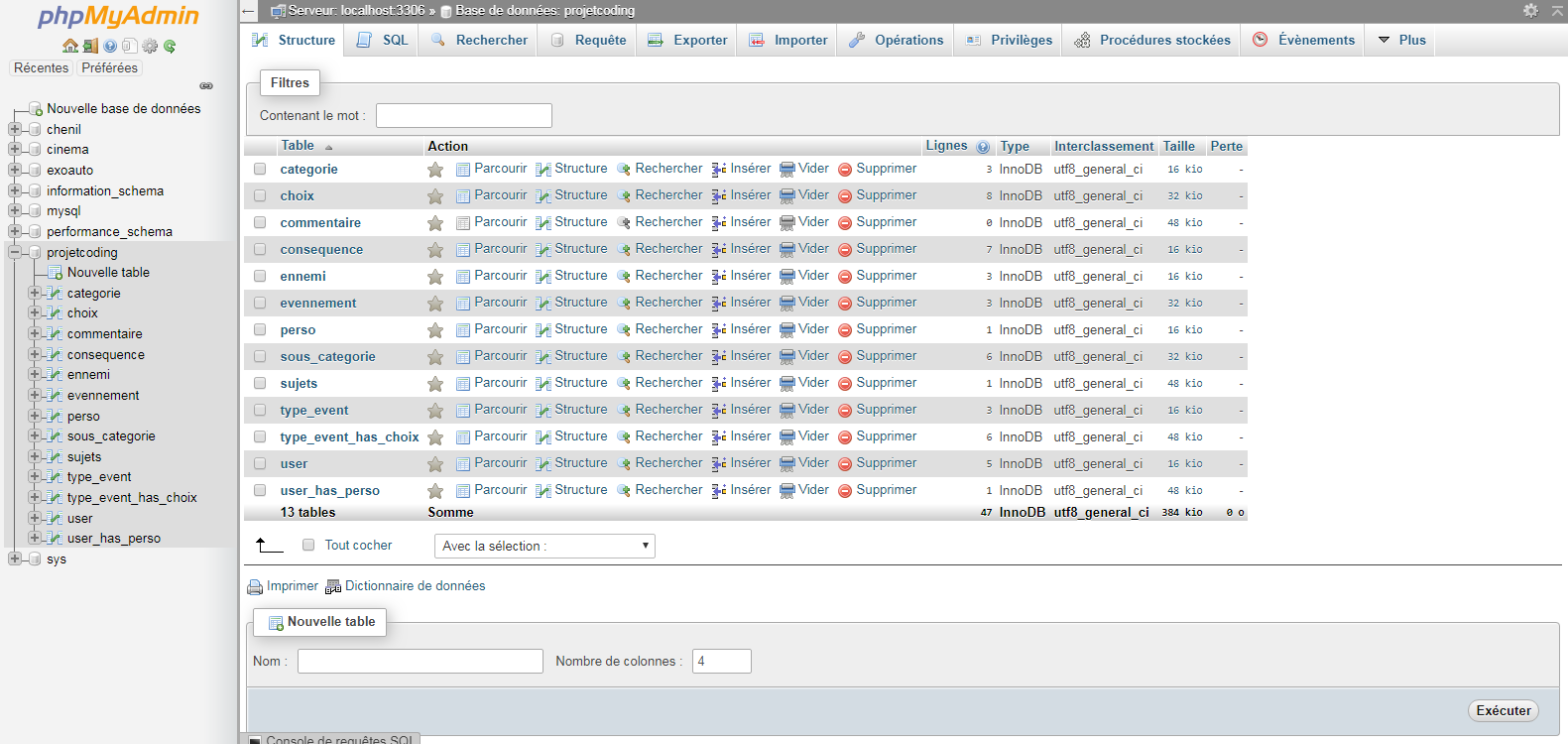
* HTML 5 qui permet de créer la structure d’un site internet
* CSS 3 qui permet d’embellir le site en y ajoutant des couleurs, des formes, des animations, …
* JavaScript qui permet d’ajouter des interactions et du dynamisme à une page web

Pour la partie back-end, j’ai utilisé les langages :

* PHP 7 qui permet d’envoyer des requêtes vers un serveur http
* MySQL qui permet d’envoyer des requêtes à une base de données

J’ai également utilisé l’application web phpMyAdmin.

Elle fournit une interface graphique permettant de visualiser et gérer plus facilement les bases de données.

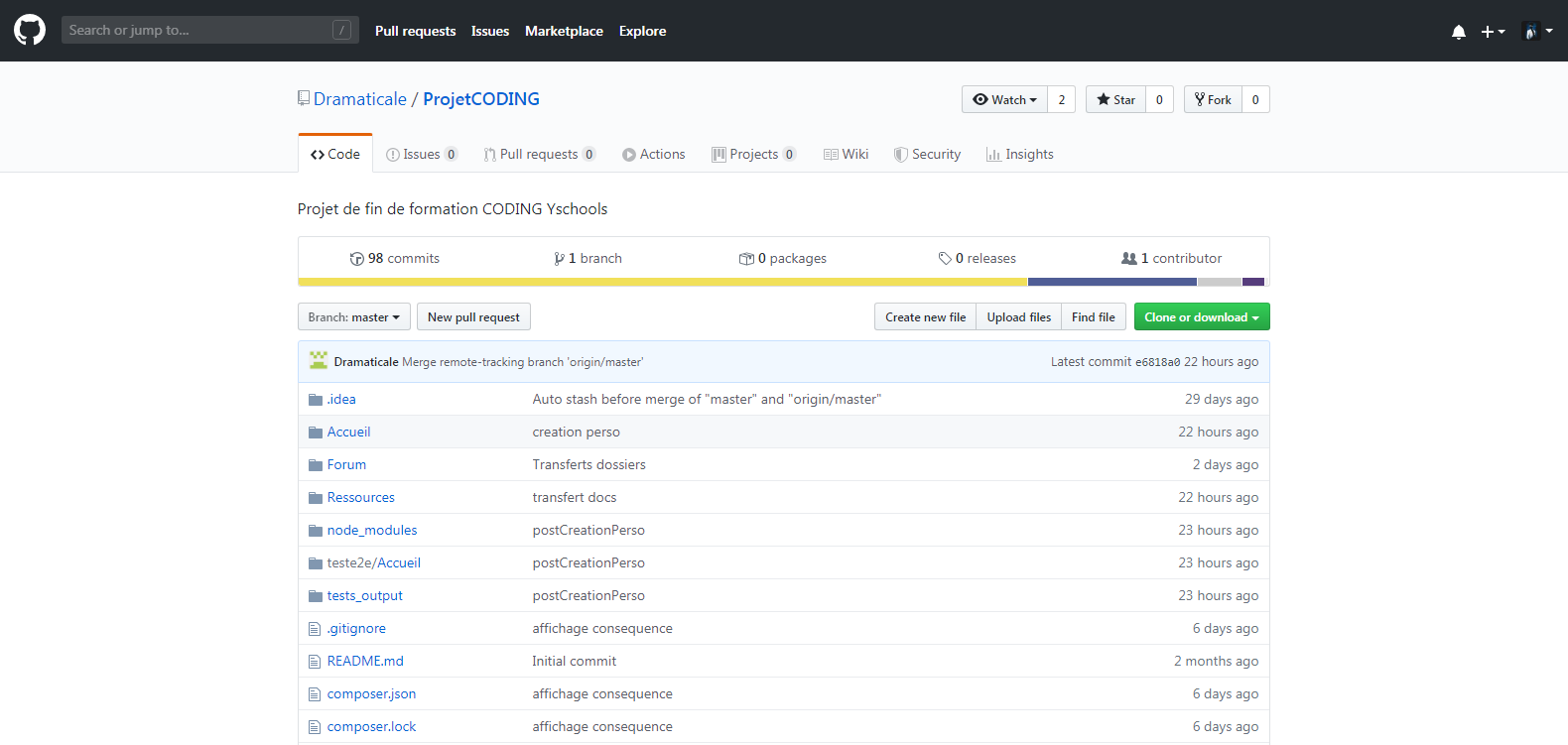


Ayant travaillé en binôme, j’ai aussi utilisé Git, GitHub et GitKraken.

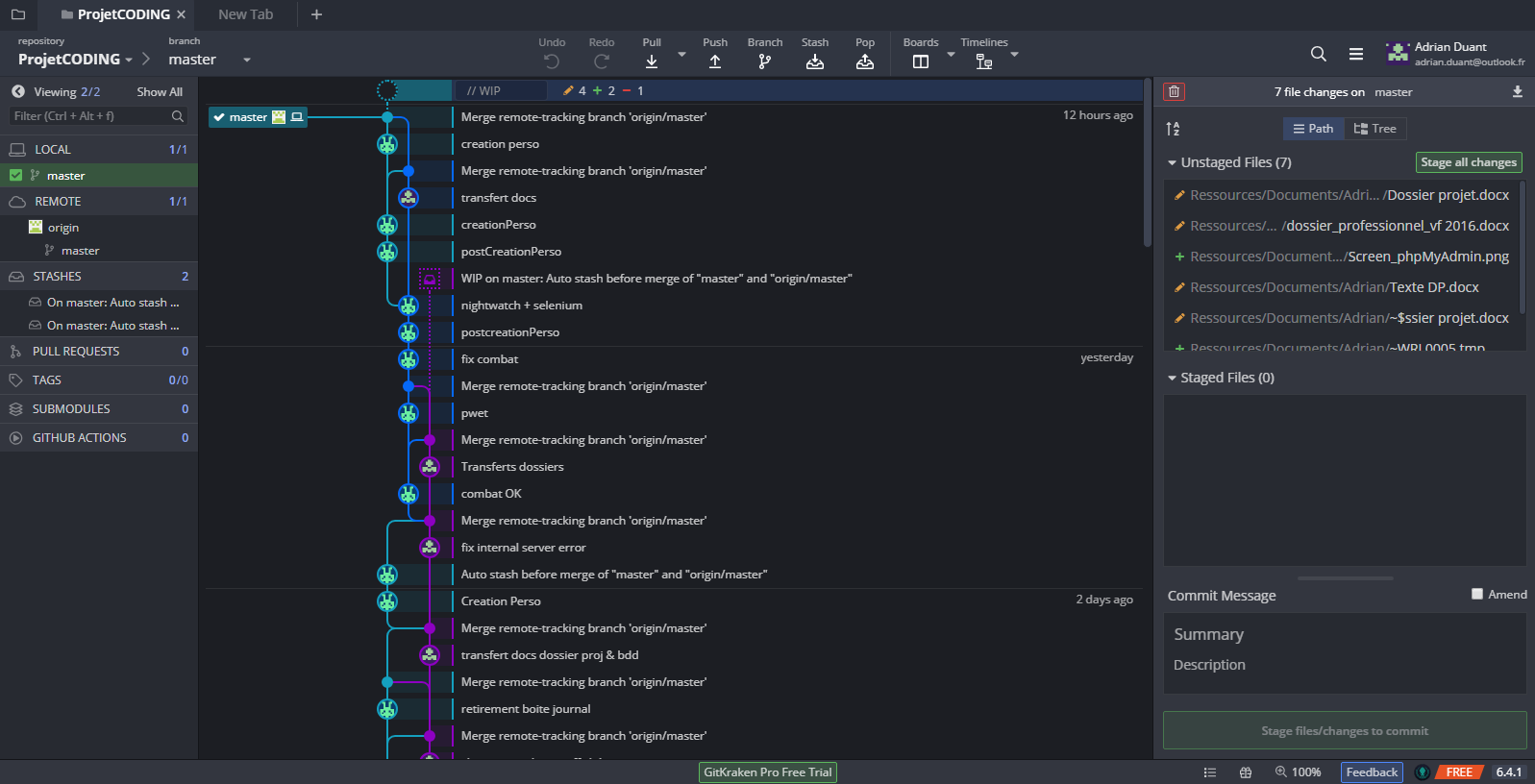
Git est un logiciel de gestion de version. Il permet de versionner un projet et ainsi pouvoir le récupérer à différent moments de son développement.

GitHub permet de mettre en ligne le projet et ainsi pouvoir le récupérer sur différentes machines.

Le projet étant partagé en totalité entre les machines, cela permet de ne pas le perdre si l’une des machines subit un problème technique important.



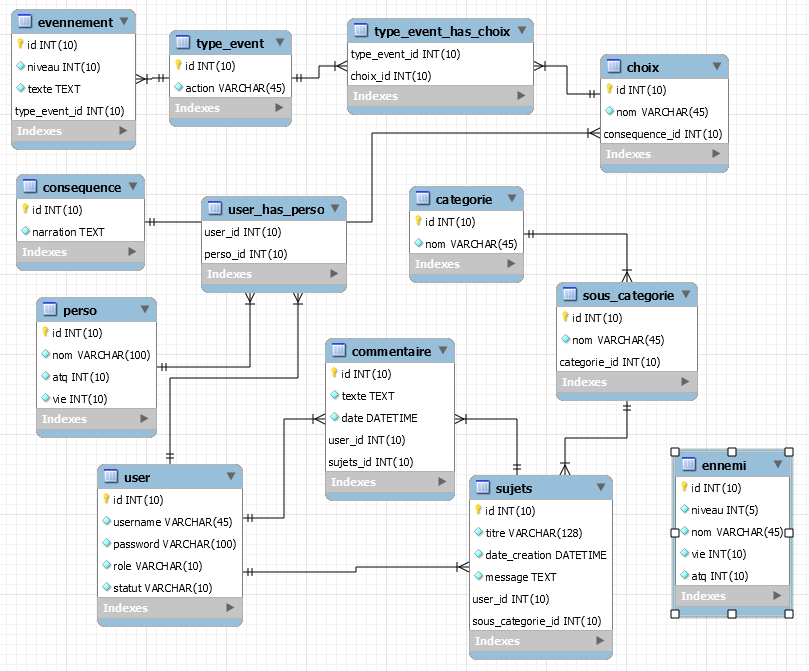
Quant à GitKraken, c’est une interface graphique pour Git. Il permet de visualiser plus simplement le versionning du projet et utiliser Git plus facilement.



Pour faire le CSS, je me suis aidé de Bootstrap pour certains éléments. Bootstrap est un framework CSS réputé pour sa simplicité d’utilisation et son côté responsive qui adapte le site en fonction la taille de l’écran de l’utilisateur.

J’ai également utilisé Nightwatch.js. Nightwatch est un framework JavaScript permettant d’effectuer des tests end-to-end en pilotant le navigateur. Cela permet de vérifier que le site web fonctionne correctement.

Base de données :



La base de données du site se présente comme ceci, mais le forum n’utilise que les tables :

* user
* categorie
* sous\_categorie
* sujets
* commentaire

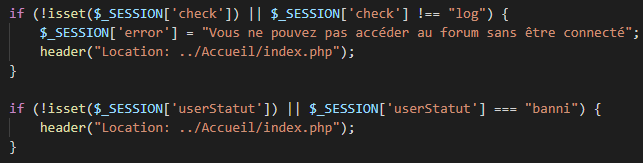
Une *sous\_categorie* correspond à une seule *categorie*, une *categorie* peut avoir plusieurs *sous\_categorie*.

Un *sujets* appartient à une *sous\_categorie* et n’est écrit que par un seul *user*, un *user* et une *sous\_categorie* peuvent avoir plusieurs *sujets* chacun.

Un *commentaire* est écrit par un seul *user* et n’appartient qu’à un seul *sujets*, un *sujets* et un *user* peuvent avoir plusieurs *commentaire* chacun.

Ici, toutes nos relations sont des one-to-many.

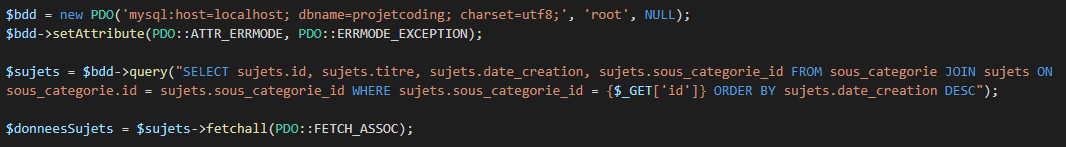
**Extraits de code significatifs :**



Cet extrait de code sert à empêcher l’accès au forum sans être connecté ou si l’utilisateur est banni.

Si l’utilisateur tente d’accéder au forum en entrant directement le lien de celui-ci dans la barre d’URL, il sera renvoyé sur la page d’accueil du site avec un message d’erreur lui disant qu’il ne peut pas accéder au forum de cette façon.

S’il tente d’accéder au forum en étant banni de celui-ci, il sera simplement renvoyé vers la page d’accueil du site.



Cet extrait de code est une requête MySQL permettant de récupérer les sujets qui seront affichés sur la page sous\_categorie.php .

Les deux premières lignes permettent de se connecter à la base de données à l’aide de PDO.

La partie SELECT de la requête demande à ce que l’on récupère l’id du sujet, son titre, sa date de création et l’id de la sous-catégorie associée.

La partie FROM de la requête indique les tables dont nous avons besoin pour cette requête. On lie ces tables à l’aide du JOIN et on indique les colonnes correspondantes avec le ON.

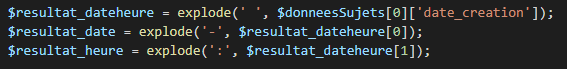
La partie WHERE indique les lignes dont on veut récupérer les données demandées avec la partie SELECT.

Ici, on sélectionne tous les sujets appartenant à la sous-catégorie dont l’id est affichée dans l’URL de la page.

La partie ORDER BY permet de définir l’ordre dans lequel les données des lignes nous sont retournées.

Ici, je souhaite que les lignes me soient retournées selon leur date de création.

Elles sont, par défaut, retournées de la plus ancienne à la plus récente et comme je souhaite qu’elles me soient retournées dans le sens inverse, j’ajoute DESC à la fin de mon ORDER BY.



On a ici une variable $donneesSujets[0][‘date\_creation’] qui contient la date de création d’un sujet récupérée de la base de données, et donc sous la forme AAAA-MM-JJ HH:MM:SS.

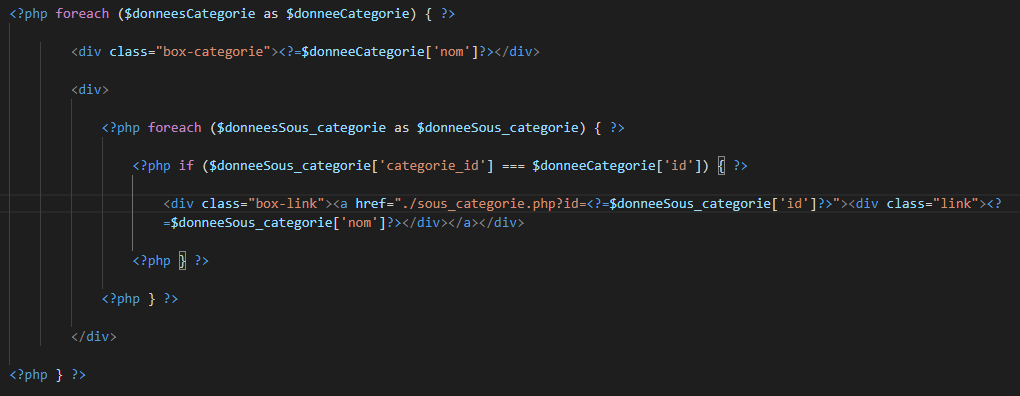
Pour pouvoir découper cette forme et afficher les différentes valeurs dans l’ordre souhaité, j’utilise la fonction explode() qui me permet de créer un tableau séparant les différentes parties de la valeur souhaitée.

Pour cette fonction, on passe en premier paramètre le caractère où notre valeur doit être séparée, puis en second paramètre la variable contenant la valeur à découper.

Ici, on commence par séparer la date et l’heure en mettant en premier paramètre ‘ ’, puis on réutilise cette fonction deux fois pour séparer l’année du mois et du jour en passant en premier paramètre ‘-’ et pour séparer l’heure des minutes et des secondes en passant en premier paramètre ‘ :’.

Cela nous donne un tableau $resultat\_date avec trois indices contenant les valeurs de la date et un tableau $resultat\_heure avec trois indices contenant les valeurs de l’heure.

On finit par afficher les valeurs dans l’ordre que l’on souhaite que l’on souhaite comme sur la dernière ligne.



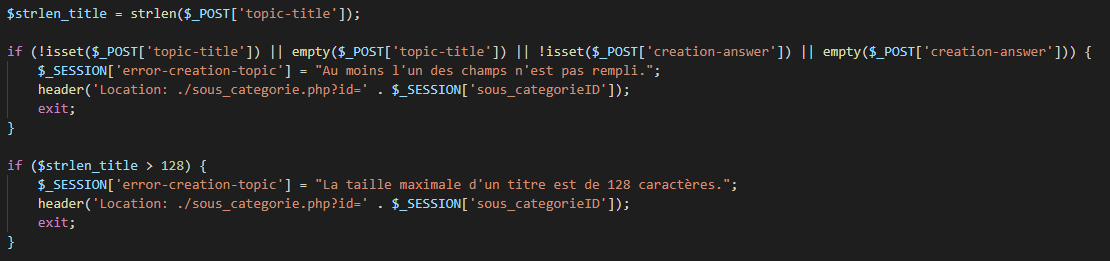
Cet extrait de code est composé de deux boucles foreach nous permettant d’afficher les catégories et leurs sous-catégories sur la page d’index du forum.

Une page appelIndex.php, inclue à cette page-ci récupère le nom et l’id de toutes les catégories et sous-catégories via une requête SQL.

Le premier foreach affiche une <div> contenant le nom de la catégorie pour chacune des catégories.

Le second foreach affiche une <div> contenant le nom et un lien d’accès à chacune des catégories, mais cette div n’est créée que si l’id de la catégorie associée à la sous-catégorie correspond à l’id de la catégorie actuellement utilisée dans le premier foreach.

On se retrouve au final avec, pour chaque catégorie, un <div> contenant son nom suivi d’une <div> pour chacune des sous-catégories associées.

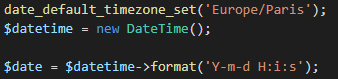


La taille d’un titre de sujet étant limitée à 127 caractères, j’ai mis sur la page postSous\_categorie.php une condition vérifiant que cette limite n’a pas été enlevée via l’inspection de la page web.

La fonction strlen() permet de récupérer le nombre de caractères de la variable entrée en paramètre si celle-ci est un string.

La première condition if vérifie que toutes les variables attendues sur cette page son bien présentes et remplies. Si elles ne le sont pas, l’utilisateur est renvoyé sur la page sous\_categorie.php avec un message d’erreur.

La seconde condition vérifie que la longueur du titre récupérée avec strlen() est bien inférieure à 127 caractères. Si elle ne l’est pas, l’utilisateur est renvoyé sur la page sous\_categorie.php avec un message d’erreur.



La première ligne de cet extrait permet de définir le fuseau horaire sur celui de Paris.

La seconde ligne récupère la date et l’heure actuelle sans problème de décalage d’horaire grâce à la fonction date\_default\_timezone\_set() précédente.

La troisième ligne met cette date sous le format AAAA-MM-DD HH:MM:SS pour pouvoir ensuite être envoyée vers la base de données sans problème.



Sur la page statuts.php, je souhaite que seul les administrateurs puissent changer les rôles.

La première condition if vérifie que le rôle de l’utilisateur voulant modifier les rôles ou/et statuts soit bien admin avant d’afficher l’input select permettant de changer les rôles.

La seconde condition if change le rôle affiché par défaut lors de l’affichage de l’input pour correspondre au rôle actuel de l’utilisateur dont on souhaite changer le rôle.

**Veille sur les failles de sécurité :**

Lors de notre formation, il nous a été demandé de nous tenir informés des failles de sécurité pouvant apparaitre sur un site web.

J’ai donc cherché sur Google les failles de sécurité les plus courante dans le développement web.

Par exemple, j’ai trouvé la faille XSS (cross-site scripting).

Cette faille consiste à entrer du code dans un site web, par exemple via un input.

Si ces données arrivent tel quel sans avoir été vérifiées, elles peuvent provoque des risques importants, comme par exemple des redirections ou du vol d’information.

Pour se protéger de cette faille, on peut utiliser la fonction htmlspecialchars() sur les variables contenant les données entrées par les utilisateurs, ce qui échappera les caractères spéciaux et empêchera l’exécution du code entré par un utilisateur malveillant.

J’ai également trouvé la faille SQLi (SQL Injection).

Cette faille consiste à modifier une requête SQL, dont la sélection est définie par des données entrées par l’utilisateur, afin de récupérer des données différentes de celles prévu à l’origine.

Le risque de cette faille est de laisser s’échapper de la base de données des informations auxquelles un utilisateur ne devrait pas avoir accès.

Pour se protéger de cette faille, on peut préparer la requête. La requête sera compilée avant d’y insérer les données entrées par l’utilisateur, ce qui empêchera l’ajout de code.

**Description d’une situation de travail ayant nécessité une recherche sur un site anglophone :**

Lors de la création de la page sous\_categorie.php du forum, j'ai voulu afficher la date de création du sujet à côté de celui-ci.

J'ai donc récupéré la date de la création du sujet via une requête SQL, l'ai découpé comme je le souhaite grâce à la fonction explode(), puis je l'ai affiché.

Le problème était que les sujets étaient affichés dans un ordre donné par l'ID, alors que je souhaitais qu'ils soient affichés du plus récent au plus ancien.

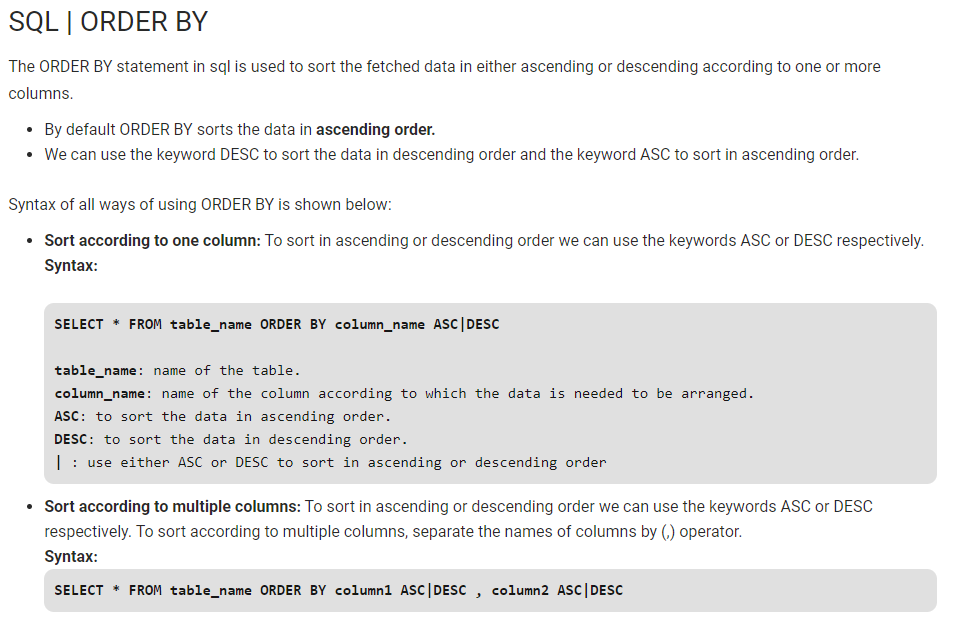
J'ai donc ajouté à ma requête SQL "ORDER BY sujets.date\_creation", ce qui affichait les sujets du plus récent au plus ancien.

Pour trouver comment inverser cet ordre d'affichage, j'ai cherché "reverse order by sql" sur Google.

Parmi les résultats, j'ai cliqué sur le premier qui semblait expliquer en détail le fonctionnement de ORDER BY.

Je suis tombé sur le site https://www.geeksforgeeks.org/sql-order-by/ où j'ai appris que, pour afficher les données retournées par une requête SQL dans un ordre descendant, il suffisait d'ajouter DESC à la fin de celle-ci.

**Traduction d’un extrait du site anglophone utilisé pour effectuer la recherche précédente :**



L'instruction ORDER BY en SQL est utilisée de sorte à retourner des données dans un sens ascendant ou descendant selon une ou plusieurs colonnes.

\* Par défaut ORDER BY trie les données dans un sens ascendant.

\* Nous pouvons utiliser le mot-clé DESC pour trier les données dans un sens descendant et le mot-clé ASC pour les trier dans un sens ascendant.

La syntaxe de toutes les façons d'utiliser ORDER BY est montrée ci-dessous :

\* Trier selon une colonne : Pour trier les données dans un sens ascendant ou descendant, nous pouvons utiliser les mots-clés ASC ou DESC respectivement.

Syntaxe :

SELECT \* FROM nom\_table ORDER BY nom\_colonne ASC|DESC

nom\_table : nom de la table

nom\_colonne : nom de la colonne selon laquelle les données ont besoin d'être arrangées

ASC : Pour trier les données dans un sens ascendant

DESC : Pour trier les données dans un sens descendant

| : Utilisez soit ASC, soit DESC pour trier dans un sens ascendant ou descendant

\* Tirer selon plusieurs colonnes : Pour trier dans un ordre ascendant ou descendant, nous pouvons utiliser les mots-clés ASC ou DESC respectivement. Pour trier selon plusieurs colonne, séparez les noms des colonnes par l’opérateur (,).

Syntaxe :

SELECT \* FROM nom\_table ORDER BY colonne1 ASC|DESC , colonne2 ASC|DESC