Remise dans le contexte :

* Freelance contact avec “les amis de l’escalade“
* Souhaite développer leur présence en ligne

(Création d’un site communautaire, but : mise en relation et partage d’information)

* Visibilité pour inciter les grimpeurs indépendants à y adhérer

Explication de l’architecture :

* Thymeleaf : (model ma vue “MVC“ (Model View Controller) model = données à afficher,
* Vue = interface graphique, contrôleur = la logique des actions effectuées par l’utilisateur, permet l’update de mon model)
* Spring boot : (qui prend en charge la majeur partie de la configuration de mon application)
* Hibernate : (pour gérer les relations entre mon application et ma base de données, gratuit et open source, orienté objet donc très facile à mettre en place avec se genre de projet)

Reprise des 10 points :

* F1 : Un utilisateur doit pouvoir consulter les informations des sites d’escalades
* F2 : Un utilisateur doit pouvoir faire une recherche à l’aide de plusieurs critères
* F3 : Un utilisateur doit pouvoir s’inscrire gratuitement sur le site.
* F4 : Un utilisateur connecté doit pouvoir partager des informations sur un site d’escalade
* F5 : Un utilisateur connecté doit pouvoir laisser des commentaires sur les pages des sites d’escalade.
* F6 : Un membre de l'association doit pouvoir taguer un site d’escalade enregistré sur la plateforme comme « Officiel Les amis de l’escalade ».
* F7 : Un membre de l'association doit pouvoir modifier et supprimer un commentaire.
* F8 : Un utilisateur connecté doit pouvoir enregistrer les topos et préciser si ces derniers sont disponibles pour être prêtés ou non.
* F9 : Un utilisateur connecté doit pouvoir consulter la liste des topos disponibles par d’autres utilisateurs et faire une demande de réservation.
* F10 : Un utilisateur connecté doit pouvoir accepter une demande de réservation. Si une réservation est acceptée, automatiquement le topo devient indisponible. L’utilisateur connecté pourra ensuite de nouveau changer le statut du topo en « disponible ».

Architecture contrôleur / service :

* Controller fait appel à ma couche service (zone de traitement des informations, calcul et retour update) qui fait appel à ma couche repository (gère les requête sql grâce à hibernate) et le repo fait appel à mon entité (“squelette“ des table dans ma BDD)

Sécurité :

* Création de profile (gestion de la sécurité de profile via des rôle), création de mot de passe
* Encrypter le Mot de passe (avec bcrypt, à la création du mot de passe la base de données détiens le mot de passe encrypter, pour vérifier le bon mot de passe on crypte le mot de passe donnée par l’utilisateur on le compare au mot de passe crypté dans notre base de données et si les deux variables sont les même alors le mot de passe donné par l’utilisateur est correct)
* Attribution des rôles (par défaut le rôle donné aux utilisateurs qui créait un compte est “USER“) mais il y a aussi le rôle “ADMIN“ donnée aux administrateurs de l’association.
* Affichage personnalisé et modification (permet d’afficher certaines informations aux utilisateurs ayant certain rôle et permet par exemple au utilisateur bénéficient du rôle d’administrateur de modifier ou supprimer des informations (comme la catégorie des commentaires))

Responcive :

* Le site est entièrement responcive (header avec bar de recherche, les tableaux)