Projet de Master

Dans ce projet, vous devez développer en groupe une application permettant de gérer des agendas. Il s'agit d'une application Web basée sur une architecture client-serveur.

Détails pratiques

Vous travaillerez par équipe de 5 ou 6 étudiants. Choisissez un nom pour votre équipe et déclarez-la sur Arche. Indiquez le lien vers le dépôt de votre projet. Si nécessaire, ajoutez les enseignants responsables en tant que membres, avec un accès au code source. Vous devrez marquer la version finale avec le tag `release`.

Cahier des charges

Vous devez développer un **unique** serveur. Celui-ci servira les pages Web et fournira l'API de votre application (distinguez les routes). Pour lancer le serveur, la commande `npm install && npm start` doit suffire. Il devra écouter le port 3000. La page Web principale sera accessible via l'adresse `htpp://localhost:3000` sur la machine hébergeant le serveur. Elle sera également visitée simultanément depuis différentes machines du réseau local.

Les données devront être persistantes d'une session à l'autre. Relancer le serveur ne doit pas provoquer de pertes d'informations. Vous n'êtes pas obligés de déployer une base de données. L'utilisation de fichiers de sauvegarde est autorisé.

Depuis le client Web, l'utilisateur doit pouvoir :
— s'identifier / fermer sa session ;
— créer un agenda ;
— ajouter/supprimer/modifier un rendez-vous ;
— visualiser un nombre quelconque d'agendas (simultanément).

Une fois ce client minimal fonctionnel, vous devrez ajouter la possibilité :
— d'ajouter des rendez-vous récurrents ;
— de rechercher un rendez-vous par différents critères ;
— de partager des agendas entre utilisateurs (ou d'annuler un partage) ;
— d'importer/exporter un agenda (format à votre convenance).

Toute idée novatrice en plus est la bienvenue.

Méthodologie

Pour chaque sprint, vous devrez pousser sur Git dans un dossier clairement identifié (comprenant le numéro du sprint) :
— le backlog du sprint courant ;
— la conception envisagée pour le sprint courant ;
— la revue du sprint précédent (fonctionnalités réalisées et validées, ou pas) ;
— la rétrospective du sprint précédent (ce qui s'est bien ou mal passé, les décisions pour le sprint courant).

À la fin de chaque sprint, il faut que la version sur la branche principale du dépôt soit à jour. Cette version doit avoir un tag correspondant au sprint actuel (v1, v2, etc). Le code doit être accompagné d'un README expliquant les aspects fondamentaux de votre projet.