Rapport d’activité hebdomadaire

Banque Populaire Val de France

GRIMAUD William

BTS Services Informatiques aux Organisations

Option Solutions Logicielles et Applications Métiers

06/01/2025 au 14/02/2025

Tuteur : Thomas DORVAL

Sommaire

Table des matières

Présentation du contexte et des missions

Présentation de l'entreprise

La Banque Populaire Val de France (BPVF) est une banque coopérative détenue par ses sociétaires. C'est une banque à taille humaine, implantée sur un vaste territoire comptant 10 départements, des Yvelines à la Vienne en passant par la Région Centre. La BPVF fait partie du groupe BPCE qui est le deuxième groupe bancaire de France. Elle est aussi actionnaire de ce même groupe.

Présentation du service Développement et Sécurité Informatique

Le service Développement et Sécurité Informatique qui appartient à la direction Système d'Information est composé de 7 personnes. Le service est divisé en 2 équipes. L'équipe de production qui va assurer le suivi de la production, faire de la maintenance corrective et évolutive des outils existants et de la gestion de projet à faible charge. L'équipe de projet va s'occuper de la gestion de projet et de la réalisation de nouveaux développements. Les deux équipes s'occupent également de faire évoluer les processus et les outils internes au service. Le service utilise principalement la technologie de développement ASP .NET CORE 6 (C#). Le service utilise aussi la méthodologie Agile et chaque matin, une réunion « Daily » va permettre à chacun de faire le point sur les travaux réalisés la veille, les difficultés rencontrées ainsi que ce qu'il a planifié pour la journée.

Présentation de l'environnement matériel et logiciel

Le service dispose de plusieurs outils logiciels :

- Microsoft Office 365 avec Outlook, Word, Excel, PowerPoint et OneNote.
- Microsoft Viva Engine qui est un réseau social d'entreprise.
- Microsoft SharePoint qui sert d'intranet et de plateforme de gestion de contenu.
- Microsoft Teams pour les réunions, la communication collaborative et la messagerie instantanée.
- Microsoft SQL Server 2019 et SQL Server Management Studio pour la gestion de base de données.
- Microsoft Visual Studio pour le développement des applications et des traitements.

- Visual TOM pour l'ordonnance de tâches.
- Teradata SQL Assistant qui permet de requêter sur les entrepôts et tables informationnelles.

Pour ce qui est de l'environnement matériel, la BPVF dispose de nombreux serveurs pour :

- La production Web IIS et les bases de données SQL
- Le développement IIS et SQL
- Les applications hors développement informatique
- La production VTOM (ordonnanceur)
- Les échanges
- L'impression

Les serveurs sont virtualisés et infogérés dans les datacenters du groupe.

Présentation des missions confiées

On m'a demandé de réaliser plusieurs améliorations sur une application existante interne au service. C'est une application appelée Portail Dev qui permet la centralisation des liens vers les applications privatives du service. Ce portail comporte aussi :

- Un module de gestion du référentiel du parc applicatif.
- Un module de gestion des tickets.
- Un module d'indicateur qualité basé sur la gestion des tickets.
- Un module de synthèse d'annuaire.
- Un module wiki.
- Un module de suivi des logs IIS (Application sur laquelle j'ai travaillé lors de mon précédent stage).
- Des schémas des architectures des serveurs d'échange de données informatisé (EDI)
- Divers liens vers des répertoires de documentations et autres ressources.

L'application utilise ASP .NET CORE 6 en C# comme langage et Microsoft SQL Server comme SGBD. L'application utilise également le modèle MVC. On m'a donné plusieurs missions à effectuer concernant cette application.

Les applications répertoriées sur le portail n'étant pas toutes privatives, on m'a tout d'abord demandé de déplacer un filtre qui permet de n'afficher que les applications privatives en cochant une case. J'ai dû déplacer cette case à cocher de la page application où sont répertoriées les applications à la page projet du portail où se trouve l'intégralité des informations concernant un projet (application concernée, base de données utilisée, traitements effectués, habilitations, niveaux DICP…).

Ensuite, il m'a été demandé d'ajouter un filtre supplémentaire sur cette même page. J'ai dû ajouter un filtre sous forme de liste déroulante pour pouvoir trier la liste des projets par les origines des application associées (Privative, prestation externe…). Une fois fait, on m'a donné deux autres filtres à ajouter sur la page projets. Une autre liste déroulante qui va trier le tableau sur l'hébergement de l'application et une autre case à cocher qui va trier le tableau sur la gestion des habilitations.

Une fois ces missions terminées, on m'a donné une autre mission sur une autre application. Cette application appelée Réservation Créneau permet de créer des événements au sein de la banque (formations, ateliers, forums…). Elle permet de créer des ateliers pour ces événements ainsi que des créneaux avec un certain nombre de places et un lieu pour ces derniers. Aussi, cette application permet aussi de s'inscrire à ces créneaux ou de signaler que l'on aurait voulu s'inscrire. Enfin, cette application possède un page appelée « Pilotage » qui permet d'effectuer un suivi des créneaux aves le nombre d'inscriptions réalisées ou impossibles et trois autres pages qui permettent de consulter le nombre d'inscriptions par atelier, fonction ou encore par direction. Cette application utilise aussi ASP .NET CORE 6 en C# comme langage et Microsoft SQL Server comme SGBD.

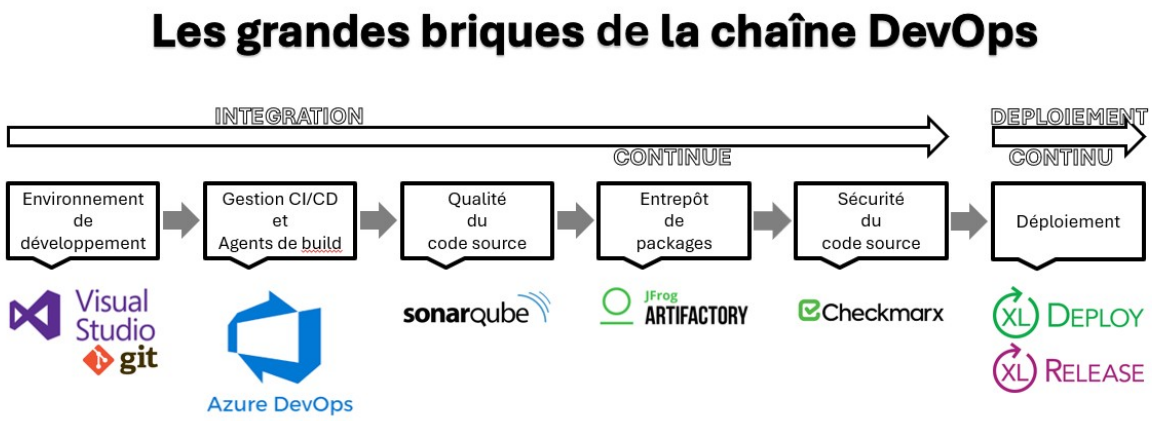
Il m'a été demandé d'ajouter un bouton sur la page de pilotage qui va permettre d'exporter toutes les statistiques concernant l'événement dans un fichier Excel qui va être composé d'une feuille pour chaque onglet de la page Pilotage plus une autre feuille pour savoir précisément qui s'est inscrit à quel créneau.

Ensuite, après avoir terminé l'export Excel, on m'a donné plusieurs tâches à effectuer sur cette même application. On m'a demandé de trier une liste déroulante de lieu sur la page de création de créneaux par ordre alphabétique et de déplacer un tooltips qui apparaît sur un champ description lors de la création d'un événement ou d'un atelier, car ce dernier cache des boutons de la barre d'outils de texte. On m'a également demandé d'ajouter une colonne sur le tableau sur la page qui affiche tous les événements pour afficher le nom de la personne qui a créé l'événement. Enfin, il fallait ajouter un bouton pour dupliquer un créneau lors de leur création pour éviter d'avoir à créer chaque créneau un par un pour chaque atelier.

Gestion de projet

Je n'ai pas reçu de date butoir pour les missions qui m'ont été confiées et elles ne sont pas divisées en sprints. L'entreprise utilisait auparavant l'outil de gestion de projet Sciforma mais il n'est plus utilisé et un autre logiciel est actuellement à l'étude pour le remplacer. Je rends compte de mon travail quotidien lors des réunion quotidiennes « Daily » et à mon tuteur. Lorsqu'un projet est terminé, une revue de code est effectuée par un membre de l'équipe. Cette phase est appelée la phase de recette. L'outil principal utilisé par le service est Azure DevOps. Cet outil va permettre la gestion de code source mais aussi de la gestion de projet. Chaque application est divisée en plusieurs branches qui vont permettre à chacun de travailler sur la même application en même temps. Il y a une branche appelée *master* qui est la branche de la version de l'application actuellement en production et il y a d'autres branches comme la branche *feature* qui elle représente les fonctionnalités en cours de développement. Cet outil dispose de tableaux de bords, de wikis, et de backlogs. Pour ce qui est du code source, chaque modification est répertoriée et il y a un système de demandes de tirage (Pull requests) qui vont permettre de faire réviser toute modification du code avant leur publication.

Tout projet est réalisé en intégration continue et le déploiement est aussi effectué en continu. Voici un schéma qui représente les grandes briques de la chaîne DevOps présente au sein du service.



L'environnement de développement est Visual Studio avec le système de contrôle de versions Git. La gestion des agents de build est effectuée par Azure DevOps. La qualité du code source est vérifiée à l'aide du logiciel SonarQube. Le service dispose d'un entrepôt de packages avec l'artifactory JFrog. La sécurité du code source est vérifiée avec l'outil Checkmarx. Enfin, le déploiement est assuré par XL Deploy et XL Release.

J'ai d'abord reçu une mission pour ajouter des filtres sur une application réalisée par le service. Je n'ai pas eu de contrainte de temps pour cette mission et je l'ai réalisée en neuf jours du mardi 7 au vendredi 17 janvier. La revue de code a été effectuée par un membre de l'équipe mardi 21 et les modifications ont été poussées sur l'application en production le même jour.

Ensuite, j'ai reçu une deuxième mission pour effectuer diverses évolutions sur une application de réservation de créneaux et de création d'événements. Je n'ai également pas eu de contrainte de temps pour cette mission. J'ai réalisé cette mission en 18 jours du 21 Janvier au 14 février. La revue de code a été effectuée par un membre de l'équipe jeudi 13 février et l'application a été mise en production le lendemain.

Rapports hebdomadaires

Semaine du 06 Janvier

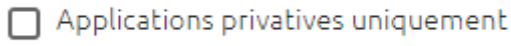
Mission 1 : Filtres portail développement et sécurité informatique

Affectation des tâches à effectuer sur une application existante. Déplacer un filtre et en ajouter d'autres pour pouvoir affiner la recherche de projets.

J'ai pris en main l'application et son code.

On m'a expliqué comment fonctionne l'application et comment fonctionne le code de manière générale. La manière dont l'application fonctionne est globalement similaire à celle de l'application sur laquelle j'ai travaillé lors de mon précédent stage. (Vues partielles pour les tableaux, javascript pour gérer les événements, spécifications pour requêter sur la base de données...).

J'ai commencé par ajouter la case à cocher sur la page des projets en reprenant celle qui était déjà présente sur la page des applications.



Ensuite, j'ai ajouté les événements javascript pour rafraichir le tableau quand on coche la case.

```
document.getElementById('isPrivateAppOnly').addEventListener('change', refreshTable);
```

```
}&showPrivateAppOnly=${isPrivateApp}`;
```

Après avoir fait cela, j'ai rencontré un problème car je devais ajouter une condition dans la spécification de la page projet pour pouvoir récupérer les applications d'origine privative.

La spécification utilisait la classe Projet mais cette classe ne contenait pas de propriété pour savoir l'origine d'une application associée à un projet. Cette propriété appartenait à la classe Application. J'ai résolu ce problème en effectuant une recherche sur internet en en m'apercevant que la classe possédait une liste d'éléments (de la classe Element).

```
25 références | 0 modification | 0 auteur, 0 modification
public virtual List<Element> Elements { get; set; }
```

Etant donné que la classe Application hérite d'Element j'ai pu faire une requête pour parcourir cette liste pour vérifier si un des éléments est une instance de la classe Application et ensuite accéder à la propriété Origine.

```
if (showPrivateAppOnly)
{
    Query.Where(e => e.Elements
        .OfType<Application>()
        .Any(app => app.Origine == Enums.Origine.Privatif));
}
```

J'ai aussi dû ajouter un paramètre supplémentaire à la spécification pour la case à cocher (ici showPrivateAppOnly). Enfin, j'ai dû rajouter ce paramètre à toutes les méthodes du contrôleur pour pouvoir le passer à la vue.

```
, bool? paramBool1 = null)
```

```
showPrivateAppOnly: paramBool1 ?? false);
```

J’ai également créé une variable dans le contrôleur qui va récupérer l’état de la case à cocher.

```
bool isPrivateAppOnly = bool.TryParse(HttpContext.Request.Query["showPrivateAppOnly"], out bool isPrivate) && isPrivate;
```

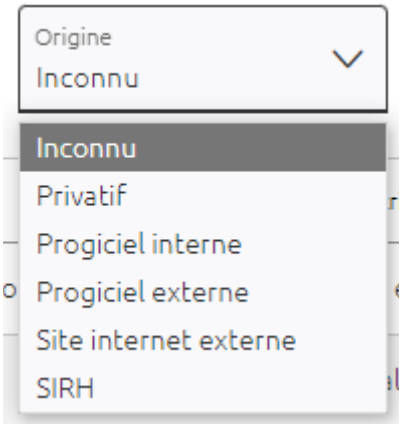
```
paramBool1: isPrivateAppOnly)
```

Une fois fait, la case était fonctionnelle et le tri s’effectuait bien sur les applications d’origine privative.

Ensuite, j’ai retiré l’ancienne case de la page application en retirant également les événements javascript reliés à cette dernière ainsi que les paramètres du contrôleur la concernant et la requête de la spécification.

```
    {
-       if (showPrivateAppOnly)
-       {
-           Query.Where(e => e.Origine == Enums.Origine.Privatif);
-       }
    }
```

Enfin, j’ai commencé à implémenter le deuxième filtre avec un liste déroulante comportant toutes les origines d’application.



Participation aux réunions quotidiennes le matin et aux vœux du Directeur Général de la BPVF pour l’année 2025.

Semaine du 13 Janvier

J’ai continué à implémenter la liste déroulante en ajoutant les événements javascript similaires à ceux implémentés précédemment.

```
const dropDownOrigine = document.getElementById('OrigineApplication');
dropDownOrigine.addEventListener('change', debounce(() => refreshTable(), 300, false));
```

```
&origineApplication=${document.getElementById('OrigineApplication').selectedOptions[0].value}
```

J'ai ajouté un paramètre et une requête dans la spécification qui va récupérer les applications dont l'origine est la même que celle sélectionnée dans la liste déroulante.

```
Origine origineApplication = Origine.Inconnu,
```

```
if (origineApplication != Origine.Inconnu)
{
    Query.Where(e => e.Elements
        .OfType<Application>()
        .Any(app => app.Origine == origineApplication));
}
```

Comme pour la case à cocher, j'ai rajouté des paramètres dans le contrôleur pour le donner à la vue.

```
Origine origineApplication = Origine.Inconnu,
```

```
origineApplication: origineApplication,
```

Ensuite, le tri marchait correctement en sélectionnant une origine via la liste déroulante.

Puis, après avoir réalisé quelques tests pour s'assurer que tout fonctionnait correctement, je me suis aperçu que quand on sélectionnait un projet puis qu'on revenait en arrière via la flèche du navigateur, la case à cocher était toujours affichée comme cochée mais que le tri n'était plus effectué et que le tableau était revenu à son état initial. J'ai commencé par aller voir sur une autre page de l'application qui comportait aussi une case à cocher pour filtrer les résultats pour vérifier si ce n'était pas intentionnel. Après m'être assuré que ce n'était pas intentionnel, j'ai regardé le code de cette autre case et je me suis rendu compte qu'elle n'avait pas été implémentée de la même manière. Pour remédier à ce problème, j'ai changé la manière dont j'avais implémenté ma case à cocher en m'inspirant de la manière dont l'autre avait été réalisée.

J'ai donc commencé par changer la façon dont étaient implémentés les événements javascript. À la place de déclarer une variable qui va regarder si la case à cocher est bien cochée dans la fonction javascript updateDataTable, je vais déclarer cette variable directement lors du chargement de la page (avec $(document).ready()) et ajouter un addEventListener exactement comme pour la liste déroulante que j'ai ajoutée précédemment ce qui donne :

```
const isPrivateApp = document.getElementById('isPrivateAppOnly').checked;
```

```
document.getElementById('isPrivateAppOnly').addEventListener('change', refreshTable);
```

Réduit à :

```
const checkboxPrivateApp = document.getElementById('isPrivateAppOnly');
checkboxPrivateApp.addEventListener('change', debounce(() => refreshTable(), 300, false));
```

J'ai aussi changé la manière dont le paramètre était implémenté ainsi que son nom dans la spécification et le contrôleur. Je l'ai changé d'un booléen nullable à un booléen classique qui va prendre l'état false par défaut :

```
, bool? paramBool1 = null)
```

Ce qui donne :

```
, bool isPrivateAppOnly = false,
```

Enfin, j'ai changé les paramètres des méthodes du contrôleur et j'ai supprimé la ligne qui récupère l'état de la case à cocher et la mettait dans une variable puisqu'elle est désormais dans un paramètre que l'on donne au contrôleur :

```
showPrivateAppOnly: paramBool1 ?? false)
```

Pour donner :

```
isPrivateAppOnly: isPrivateAppOnly,
```

Après avoir terminé d'ajouter les deux filtres et résolu ce problème, on m'a donné deux autres filtres à implémenter sur cette même page. D'abord un autre filtre sous forme de liste déroulante cette fois sur l'hébergement de l'application et un autre filtre sous forme de case à cocher pour trier les applications qui ne sont pas sous un certain système de gestion des droits.

Pour la liste déroulante, vu que la classe Application possède aussi une propriété qui appartient à une classe Enumerable pour l'hébergement de l'application, le principe est le même que pour l'implémentation de la liste de l'origine de l'application. La requête est aussi la même, on change juste la propriété qu'on veut récupérer.

```
if (hebergementApplication != Hebergement.Inconnu)
{
    Query.Where(e => e.Elements
        .OfType<Application>()
        .Any(app => app.Hebergement == hebergementApplication));
}
```

On rajoute les paramètres au contrôleur :

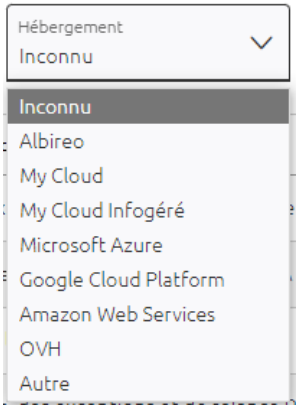
```
Hebergement hebergementApplication = Hebergement.Inconnu,
```

```
hebergementApplication: hebergementApplication,
```

Enfin, j'ai déclaré une nouvelle variable pour l'événement javascript :

```
const dropDownHebergement = document.getElementById('HebergementApplication');
dropDownHebergement.addEventListener('change', debounce(() => refreshTable(), 300, false));
```

Voici le résultat complet :

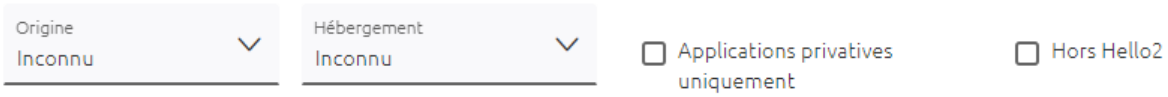


Pour la case à cocher, idem, la gestion des droits étant aussi une propriété Enumerable de classe Application.

```
if (isNotHello2)
{
    Query.Where(e => e.Elements
        .OfType<Application>()
        .Any(app => !(app.GestionDroits == Enums.GestionDroits.Hello2)));
}
```

Ici, on veut récupérer toutes les applications qui ne sont pas sous Hello2.

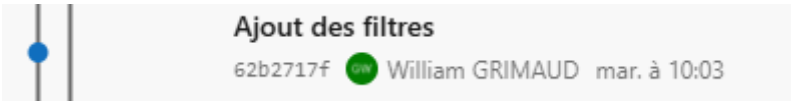
Voici le résultat de la barre avec tous les filtres que j'ai implémentés :

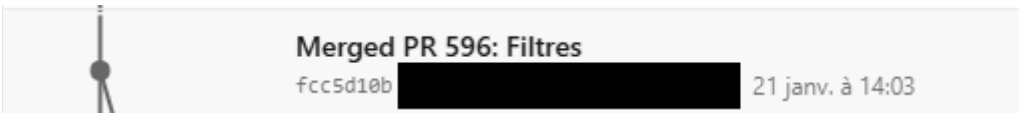


Participation aux réunions quotidiennes le matin où je rends compte de l'avancée de mon travail, de potentiels problèmes rencontrés et de ce je compte faire dans la journée. Ces réunions peuvent m'aider dans mon travail. Je peux demander l'avis ou de l'aide à des membres de l'équipe pour résoudre un problème. J'ai aussi participé à une plénière toute la journée de mardi. Cette plénière est organisée annuellement et tous les services appartenant à la direction du système d'information sont présents. Il y a eu une présentation de chaque service par ses membres, des intervenants et des ateliers concernant la satisfaction collaborateur en groupe et une présentation de la feuille de route pour l'année 2025.

Semaine du 20 Janvier

Ayant terminé d'ajouter les filtres sur le portail, j'ai créé une branche feature et j'ai poussé mes modifications pour qu'un membre de l'équipe vérifie qu'il n'y ait pas de problèmes. Je n'ai pas eu de problèmes pendant la revue de code et une demande de tirage a été effectuée puis les modifications ont été mises sur l'application en production.

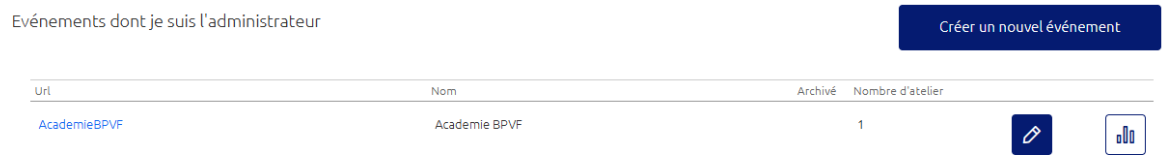




Une fois cela terminé, on m'a donné une autre mission à réaliser sur une autre application de réservation de créneaux pour réaliser un export Excel (voir section présentation des missions confiées).

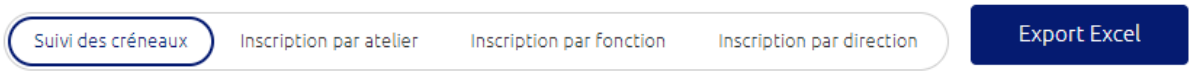
Mission 2 : évolutions sur l'application Réservation Créneau

Page d'accueil avec tous les événements dont je suis l'administrateur :



A droite sont les boutons pour modifier l'événement et pour accéder à la page pilotage où va se trouver le bouton de l'export Excel.

J'ai donc commencé par ajouter le bouton qui va permettre l'export.



J'ai ensuite commencé à écrire le code Javascript nécessaire avec les événements JQuery et le nom du fichier Excel mais j'ai rencontré un problème car je ne savais pas comment récupérer l'id de l'événement pour le donner au contrôleur. Un membre de l'équipe m'a donc suggéré de créer un champ input de type hidden pour récupérer cet id.

```
<input id="idEvenement" name="idEvenement" type="hidden" value="@Model.Id" />
```

Puis j'ai complété le code javascript avec cette valeur.

```
$(document).on('click', '#BtnExport', async function () {
    let evenement = $('#idEvenement').val();

    const response = await fetch('@Url.Action("ExportStatsEvenement", "Evenement")?id=' + encodeURIComponent(evenement));
    if (response.ok) {
        return response.blob()
            .then(blob => {
                var url = window.URL.createObjectURL(blob);
                var a = document.createElement('a');
                a.href = url;
                a.download = 'Stats';
                document.body.appendChild(a);
                a.click();
                a.remove();
            });
    } else {
        return response.text()
            .then(res => {
                console.error('Error:', res);
                changeToasterState89C3("Une erreur s'est produite", 'bpce-toaster-loader-error', true, false);
            })
    }
});
```

Cela étant fait, j'ai créé une nouvelle méthode dans le contrôleur qui va donner l'id de l'événement à la méthode du service pour l'export.

```
[HttpGet]
0 références | 0 modification | 0 auteur, 0 modification
public async Task<IActionResult> ExportStatsEvenementAsync(string id)
    => await _exportService.ExportStatsParEvenementAsync(id);
```

Enfin, j'ai créé un nouveau dossier pour ajouter le service pour l'export des données. Dans ce fichier, j'ai créé la méthode ExportStatsParEvenementsAsync qui va créer le fichier Excel ainsi que toutes les colonnes, feuilles et va l'alimenter avec les données voulues.

Pour réaliser cette méthode la personne qui m'a donné cette mission m'a dit d'utiliser la bibliothèque ClosedXML qui est nécessaire pour la manipulation et la création de fichiers Excel. On m'a aussi donné une autre application où était réalisé un export Excel pour avoir quelques bases.

Tout d'abord, j'ai commencé par créer une liste de tous les ateliers de l'événement passé en paramètre avec une méthode déjà existante.

```
List<AtelierViewModel> ateliers = await _evenementService.GetAllAteliersByIdEvenementAsync(id);
```

Ensuite, j'ai créé quatre variables de type DataTable qui vont comporter le contenu de chaque feuille du fichier.

```
DataTable dataTableSuivi = new();
DataTable dataTableInscription = new();
DataTable dataTableInscriptionAtelier = new();
DataTable dataTableInscriptionFonction = new();
```

Après les avoir créées, il faut créer toutes les colonnes nécessaires qui vont apparaitre dans le fichier Excel.

```
// Entête des colonnes
dataTableSuivi.Columns.Add("Atelier");
dataTableSuivi.Columns.Add("Créneau");
dataTableSuivi.Columns.Add("Inscription");
dataTableSuivi.Columns.Add("Inscription impossible");

dataTableInscription.Columns.Add("Atelier");
dataTableInscription.Columns.Add("Date début");
dataTableInscription.Columns.Add("Heure début");
dataTableInscription.Columns.Add("Heure fin");
dataTableInscription.Columns.Add("Participant");
dataTableInscription.Columns.Add("Fonction");

dataTableInscriptionAtelier.Columns.Add("Atelier");
dataTableInscriptionAtelier.Columns.Add("Inscriptions");
dataTableInscriptionAtelier.Columns.Add("Inscription impossible");

dataTableInscriptionFonction.Columns.Add("Atelier");
dataTableInscriptionFonction.Columns.Add("Fonction");
dataTableInscriptionFonction.Columns.Add("Inscriptions");
```

Un fois fait, j'ai alimenté le fichier Excel en remplissant les colonnes avec les données demandées. Pour cela, j'ai effectué plusieurs boucles sur toutes les collections des classes que comporte un événement. La classe événement possède une liste d'atelier qui possède une liste de créneaux qui possède une liste d'inscriptions. J'ai donc créé 3 boucles en partant de la variables ateliers créés précédemment. Dans chacune de ces boucles, j'ai alimenté la feuille correspondante avec les données dont on avait besoin. Par exemple, la feuille concernant le nombre d'inscriptions par atelier se situait uniquement dans la première boucle car je n'avais besoin que de propriétés appartenant à la classe Atelier.

```
foreach (var atelier in ateliers)
{
    dataTableInscriptionAtelier.Rows.Add(
        atelier.Titre,
        atelier.CreneauxViewModel.SelectMany(creneau => creneau.InscriptionsViewModel).Count(inscription => inscription.Valide),
        atelier.CreneauxViewModel.SelectMany(creneau => creneau.InscriptionsViewModel).Count(inscription => !inscription.Valide)
    );
```

Ici, dataTableInscriptionAtelier va donner cette feuille dans le fichier Excel :

Atelier	Inscription	Inscription impossibl
Séparation des univers Particulier et Professionnel sous Cyber ? Pourquoi et comment ?	1	1
Accompagnement du déploiement des attributs GREEN (Vertuose)	3	0
Le parcours médical après souscription d'un contrat de prévoyance pro	0	0
Enveloppe financement BEI Santé	3	0
Enveloppe financement BEI Santé	0	2

J'ai répété cette opération pour deux autres feuilles puis j'ai eu un problème lors de l'alimentation de la feuille qui répertorie le nombre d'inscriptions par fonction. En effet, toutes les fonctions étaient affichées dans le fichier. Par exemple, si un chef de projet s'était inscrit à un atelier, cela affichait un compte pour la fonction chef de projet sur les autres ateliers, même si aucune personne ayant cette fonction ne s'était inscrit à cet atelier (le compte était à 0). J'ai dû créer une autre liste qui elle ne va comporter que les fonctions uniques qui se sont inscrites à un atelier pour la parcourir et récupérer toutes les fonctions.

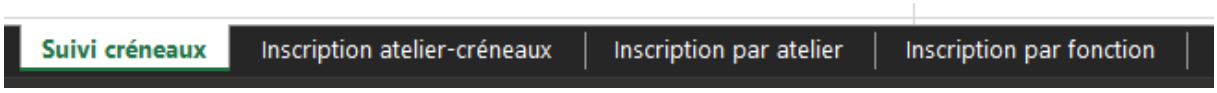
```
var fonctionsAvecInscriptions = atelier.CreneauxViewModel.SelectMany(creneau => creneau.InscriptionsViewModel).Where(inscription => inscription.Valide).Select(inscription =>

foreach (var fonction in fonctionsAvecInscriptions)
{
    dataTableInscriptionFonction.Rows.Add(
        atelier.Titre,
        fonction,
        atelier.CreneauxViewModel.SelectMany(creneau => creneau.InscriptionsViewModel).Count(inscription => inscription.Fonction == fonction && inscription.Valide)
    );
}
```

Pour terminer cette méthode, j'ai ajouté une liste de feuilles avec toutes les données ajoutées précédemment pour les créer et leur donner un nom puis j'ai utilisé la méthode de la bibliothèque pour générer un fichier Excel comme valeur à retourner.

```
List<Tuple<DataTable, string>> sheets = new()
{
    new Tuple<DataTable, string>(dataTableSuivi, "Suivi créneaux"),
    new Tuple<DataTable, string>(dataTableInscription, "Inscription atelier-créneaux"),
    new Tuple<DataTable, string>(dataTableInscriptionAtelier, "Inscription par atelier"),
    new Tuple<DataTable, string>(dataTableInscriptionFonction, "Inscription par fonction")
};

return GenerateExcelFile(sheets);
```



Comme chaque semaine, j'ai participé aux réunions quotidiennes le matin.

Semaine du 27 Janvier

Ensuite, sur la feuille Inscription atelier-créneaux, j'ai eu un problème pour alimenter la colonne qui récupère l'entité de la personne inscrite. En effet, je n'arrivais pas à récupérer l'entité car ce n'était pas une propriété de la classe InscriptionViewModel. J'ai demandé de l'aide à un membre de l'équipe et on m'a suggéré de créer une liste de la classe Collaborateur qui comportait cette propriété. Par ailleurs, les propriétés de cette classe sont directement récupérées dans la base de données de l'annuaire donc s'il y a un changement de nom ou d'entité, il sera mis à jour automatiquement dans l'application.

```
List<Collaborateur> participants = await _collaborateurService.GetCollabByIdsCollabsAsync(evenement.AteliersViewModel
    .SelectMany(atelier => atelier.CreneauxViewModel)
    .SelectMany(creneau => creneau.InscriptionsViewModel)
    .Select(x => x.IdCollab)
    .ToList());
```

J'ai ensuite utilisé ces propriétés pour les afficher dans le fichier Excel tout en m'assurant que l'application ne plante pas si jamais une de ces valeurs est nulle en ajoutant un opérateur conditionnel null.

```
dataTable.Rows.Add(
    atelier.Titre,
    creneau.DateDebut.ToString("dd/MM/yyyy"),
    creneau.DateDebut.ToString("HH'h'mm"),
    creneau.DateFin.ToString("HH'h'mm"),
    participants?.FirstOrDefault(x => x.IdCollab == inscription.IdCollab)?.Libelle,
    participants?.FirstOrDefault(x => x.IdCollab == inscription.IdCollab)?.Entite,
    participants?.FirstOrDefault(x => x.IdCollab == inscription.IdCollab)?.Fonction
```

Après avoir terminé, d'alimenter le fichier Excel, on m'a demandé de factoriser mon code pour plus de clarté. J'ai donc créé des méthodes privées pour l'initialisation des colonnes, l'alimentation des colonnes et la création des feuilles.

```
private DataTable CreateDataTableInscription()
{
    var dataTable = new DataTable();
    dataTable.Columns.Add("Atelier");
    dataTable.Columns.Add("Date début");
    dataTable.Columns.Add("Heure début");
    dataTable.Columns.Add("Heure fin");
    dataTable.Columns.Add("Participant");
    dataTable.Columns.Add("Entité");
    dataTable.Columns.Add("Fonction");
    return dataTable;
}
```

```
private void PopulateDataTableInscription(DataTable dataTable, AtelierViewModel atelier, List<Collaborateur> partic
{
    foreach (var creneau in atelier.CreneauxViewModel)
    {
        foreach (var inscription in creneau.InscriptionsViewModel.Where(inscription => inscription.Valide))
        {
            dataTable.Rows.Add(
                atelier.Titre,
                creneau.DateDebut.ToString("dd/MM/yyyy"),
                creneau.DateDebut.ToString("HH'h'mm"),
                creneau.DateFin.ToString("HH'h'mm"),
                participants?.FirstOrDefault(x => x.IdCollab == inscription.IdCollab)?.Libelle,
                participants?.FirstOrDefault(x => x.IdCollab == inscription.IdCollab)?.Entite,
                participants?.FirstOrDefault(x => x.IdCollab == inscription.IdCollab)?.Fonction
            );
        }
    }
}
```

```
private List<Tuple<DataTable, string>> CreateSheets(params DataTable[] dataTables)
{
    var sheets = new List<Tuple<DataTable, string>>
    {
        new Tuple<DataTable, string>(dataTables[0], "Suivi créneaux"),
        new Tuple<DataTable, string>(dataTables[1], "Inscription atelier-créneaux"),
        new Tuple<DataTable, string>(dataTables[2], "Inscription par atelier"),
        new Tuple<DataTable, string>(dataTables[3], "Inscription par fonction")
    };

    return sheets;
}
```

Enfin, dans la méthode principale j'utilise ces fonctions pour alimenter le fichier Excel.

```
public async Task<FileStreamResult> ExportStatsParEvenementAsync(string id)
{
    EvenementViewModel evenement = await _evenementService.GetEvenementByIdAsync(id);

    DataTable dataTableSuivi = CreateDataTableSuivi();
    DataTable dataTableInscription = CreateDataTableInscription();
    DataTable dataTableInscriptionAtelier = CreateDataTableInscriptionAtelier();
    DataTable dataTableInscriptionFonction = CreateDataTableInscriptionFonction();

    foreach (var atelier in evenement.AteliersViewModel)
    {
        PopulateDataTableInscriptionAtelier(dataTableInscriptionAtelier, atelier);
        await PopulateDataTableInscriptionAsync(dataTableInscription, atelier);
        await PopulateDataTableInscriptionFonctionAsync(dataTableInscriptionFonction, atelier);
        await PopulateDataTableSuiviAsync(dataTableSuivi, atelier);
    }

    List<Tuple<DataTable, string>> sheets = CreateSheets(dataTableSuivi, dataTableInscription, dataTableInscriptionAtelier, dataTableInscriptionFonction);

    return GenerateExcelFile(sheets);
}
```

Une fois terminé, j'ai poussé mon travail sur cet export et j'ai créé une demande de tirage. Après une revue de mon code, un membre de l'équipe m'a demandé d'effectuer des corrections. Certaines méthodes qui alimentent les colonnes sont de type async mais la méthode s'exécutait de manière synchrone donc ce type était incorrect. J'ai donc changé le type en void. Pour terminer, la liste de collaborateur était déclarée dans une boucle ce qui, lors de gros événements avec de nombreux ateliers et créneaux va effectuer un grand nombre de requêtes en base de données. On m'a donc demandé de charger l'ensemble des collaborateurs une fois (2000 collaborateurs) plutôt que 50*10 collaborateurs par exemple.

J’ai déplacé ma liste de collaborateur dans la méthode principale et je l’ai fournie en paramètre uniquement aux méthodes qui vont en avoir besoin.

Résultat final :

```
public async Task<FileStreamResult> ExportStatsParEvenementAsync(string id)
{
    EvenementViewModel evenement = await _evenementService.GetEvenementByIdAsync(id);

    List<Collaborateur> participants = await _collaborateurService.GetCollabByIdsCollabsAsync(evenement.AteliersViewModel
        .SelectMany(atelier => atelier.CreneauxViewModel)
        .SelectMany(creneau => creneau.InscriptionsViewModel)
        .Select(x => x.IdCollab)
        .ToList());

    DataTable dataTableSuivi = CreateDataTableSuivi();
    DataTable dataTableInscription = CreateDataTableInscription();
    DataTable dataTableInscriptionAtelier = CreateDataTableInscriptionAtelier();
    DataTable dataTableInscriptionFonction = CreateDataTableInscriptionFonction();

    foreach (var atelier in evenement.AteliersViewModel)
    {
        PopulateDataTableInscriptionAtelier(dataTableInscriptionAtelier, atelier);
        PopulateDataTableInscription(dataTableInscription, atelier, participants);
        PopulateDataTableInscriptionFonction(dataTableInscriptionFonction, atelier);
        PopulateDataTableSuivi(dataTableSuivi, atelier);
    }

    List<Tuple<DataTable, string>> sheets = CreateSheets(dataTableSuivi, dataTableInscription, dataTableInscriptionAtelier, dataTableInscriptionFonction);

    return GenerateExcelFile(sheets);
```

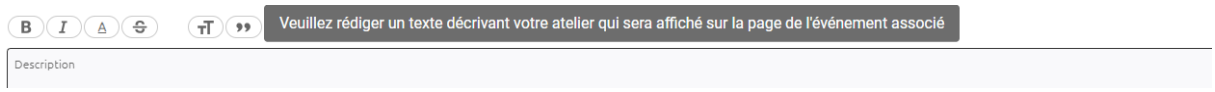
J’ai ensuite poussé ces corrections et toutes mes évolutions ont été mises sur l’application en production.

Après avoir terminé l’export Excel, on m’a donné d’autres évolutions à effectuer sur cette application : trier la liste déroulante des lieux des créneaux, déplacer un tooltips gênant, afficher le nom du créateur d’un événement sur le tableau général des événements et ajouter un bouton pour dupliquer les créneaux lors de leur création.

J’ai commencé par trier la liste des lieux d’un événement. J’ai simplement ajouté une instruction OrderBy sur le libelle du lieu dans la méthode qui va récupérer la liste des lieux dans la base de données.

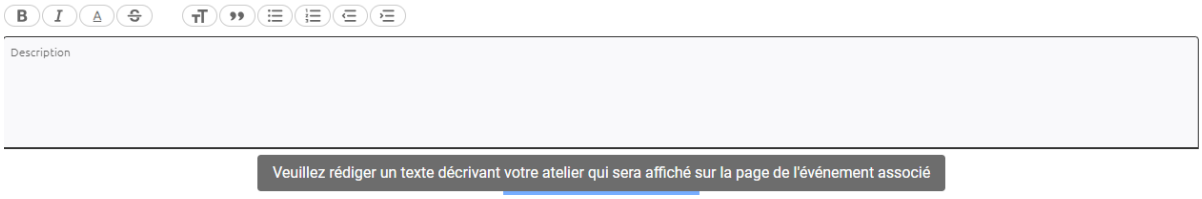
```
return listLieux
    .Where(x => x.IsArchive == false)
    .Select(x => new ListLieuViewModel() { Id = (int)x.Id, Libelle = (x.Salle != null ? x.Libelle + " - " + x.Salle : x.Libelle), Salle = x.Salle })
    .OrderBy(x => x.Libelle)
    .ToList();
```

Ensuite, j’ai déplacé le tooltips qui obstruait la barre d’outils de texte sur le champ description sur les pages de création d’un événement et d’un atelier.



Ici, le tooltips bloque l’accès à des boutons de la barre d’outils pour mettre en forme le texte. En cherchant dans le code dans le projet, j’ai trouvé un attribut qui permet de changer la position de ce tooltips et je l’ai donc positionné en dessous du champ description plutôt qu’au-dessus.

```
data-mdb-placement="bottom"
```



J'ai donc ensuite commencé à travailler sur l'affichage du nom du créateur dans la liste des événements. La personne qui m'a assigné cette mission m'a dit qu'il serait nécéssaire d'ajouter une colonne pour savoir si l'administrateur de l'événement est également son créateur dans la table d'administration d'un événement dans la base de données. L'application utilisant Entity Framework Core et le service utilisant une démarche Code First, pour ajouter cette colonne, il faut réaliser une migration de la base de données. Pour cela, un membre de l'équipe m'a montré la démarche à suivre. Tout d'abord, il faut ajouter la propriété que l'on souhaite ajouter dans la classe concernée (ici EvenementAdmin). Cette propriété est un booléen non nullable qui va prendre en valeur par défaut false.

```
2 références | 0 modification | 0 auteur, 0 modification
public bool IsCreateur { get; set; }
```

Ensuite, il faut répéter cet ajout dans le ViewModel associé à cette classe. Puis, il faut ajouter cette propriété dans le fichier de mapping de la classe EvenementAdmin pour que la migration soit réalisée.

```
builder.Property(x => x.IsCreateur).IsRequired().HasDefaultValue(false);
```

Enfin, il faut rentrer une série de commandes dans la console du gestionnaire de package pour créer la migration et mettre à jour la base de données.

D'abord commencer par affecter la variable d'environnement à l'environnement local :

```
PM> $env:ASPNETCORE_ENVIRONMENT = 'Local'
```

Créer la migration :

```
PM> add-migration NomDeLaMigration -context NomDuContexteDeLaBD
```

Mettre à jour la base de données avec la nouvelle colonne :

```
PM> update-database -context NomDuContexteDeLaBD
```

La colonne IsCreateur est maintenant ajoutée dans la table EvenementAdmin de la base de données :

IsCreateur
1

Ensuite, j'ai modifié la vue du tableau des évènements pour y ajouter la colonne avec le nom du créateur.

```
@{
    var createur = evenement.EvenementAdminsViewModel.FirstOrDefault(x => x.IsCreateur);

    if (createur != null)
    {
        <td>@createur.Collaborateur.Libelle</td>
    }
    else
    {
        <td></td>
    }
}
```

J'ai eu un problème par la suite car la variable evenement est une liste de type EvenementViewModel et que cette classe contient une liste de EvenementAdminViewModel. Cette liste était vide lors de la récupération des événements et que ma variable createur qui contient tous les collaborateurs qui ont créé un événement était donc vide. J'ai ensuite remarqué avec un membre de l'équipe que dans la méthode du service qui récupère tous les événements, lorsque l'utilisateur connecté avait le rôle administrateur (ce qui était mon cas), une liste vide était renvoyée. J'ai donc modifié cette condition.

```
var evenements = _webSession.CurrentUser.IsInRole(Role.Administrateur) ?
    await _evenementRepository.ListAsync() :
    await _evenementRepository.ListAsync(new EvenementsByCollabSpec(_webS
```

```
var evenements = await _evenementRepository.ListAsync(new EvenementsByCollabSpec(!_webSession.CurrentUser.IsInRole(Role.Administrateur) ?
                                                                                   _webSession.CurrentUser.IdCollab :
                                                                                   null));
```

C'est dans la spécification EvenementsByCollabSpec que se trouvait l'instruction Include qui va charger la classe EvenementAdmin. Donc elle n'était pas chargée si mon rôle est administrateur.

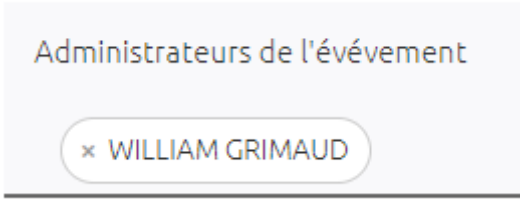
```
Query
    .Include(x => x.EvenementAdmins);
```

J'ai également dû ajouter le mapping pour que Evenement ait pour membre EvenementAdmin.

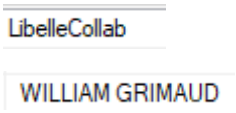
```
config.CreateMap<EvenementViewModel, Evenement>()
    .ForMember(x => x.Ateliers, opt => opt.MapFrom(src => src.AteliersViewModel))
    .ForMember(x => x.EvenementAdmins, opt => opt.MapFrom(src => src.EvenementAdminsViewModel))
    .ReverseMap();
```

Après avoir réglé ce problème, j'ai rencontré un autre problème car dans EvenementAdminViewModel se trouvait une propriété Collaborateur. Or, même problème, cette propriété était vide. J'ai demandé de l'aide à un membre de l'équipe car je n'arrivais pas à trouver d'où venait cette erreur et on a remarqué que Collaborateur était stockée dans une vue car que cette table appartenait à la base de données de l'annuaire. Donc, lorsque j'ai essayé d'utiliser l'instruction Include dans la même spécification vue précédemment pour ajouter Collaborateur, une erreur est survenue me disant que l'objet Observateur était introuvable. Pour régler ce problème, le membre de l'équipe qui était venu m'aider m'a dit qu'il serait plus simple d'ajouter une colonne dans la table EvenementAdmin qui contient le

nom du collaborateur qui a été ajouté dans la liste des administrateurs lors de la création d'un événement.



J'ai donc créé une autre migration de la même manière que pour la première.



Une fois chose faite, j'ai dû modifier les méthodes du service qui fournissait la liste des événements à la vue pour y ajouter le nom du créateur.

```
IdEvenement = viewModel.Id, IdCollab = _webSession.CurrentUser.IdCollab, LibelleCollab = _webSession.CurrentUser.Libelle, IsCreateur = true
```

J'ai aussi remarqué que quand le collaborateur qui était connecté créait un événement, il était automatiquement mis en tant qu'administrateur (sans l'afficher dans la liste) de l'événement et que si on ajoutait cette personne manuellement dans la liste des administrateurs, cela créait un doublon dans la base de données. J'ai donc ajouté une instruction qui va le retirer s'il se met dans la liste.

```
.IdCollaborateursList.Where(x => x != _webSession.CurrentUser.IdCollab.ToString("0000000"))
```

Enfin, j'ai dû faire en sorte que l'utilisateur qui créait l'événement était désigné comme le créateur de ce dernier et donc que la colonne IsCreateur dans la base de données soit mise à true.

```
IsCreateur = true
```

Après avoir apporté toutes ces modifications, le nom du créateur de l'événement apparaissait dans le tableau.

Comme chaque semaine, j'ai participé aux réunions quotidiennes le matin.

Semaine du 3 Février

Après avoir terminé d'ajouter le nom du créateur de l'évènement, j'ai commencé à implémenter la duplication des créneaux.

J'ai commencé par ajouter le bouton qui va permettre cette duplication sur la page de création d'un créneau.

J'ai ajouté une condition à ce bouton pour qu'il n'apparaisse pas lorsque l'on veut modifier un créneau et non le créer (la page pour les deux étant la même).

```
<div class="col-2 mt-2">
    @if (Model.IsNew)
    {
        <button type="button" class="bpce-button bpce-button-primary bpce-button-no-border bpce-button-icon-only"
    }
    else
    {
        <span></span>
    }
</div>
```

J'ai également changé le model de la vue CreneauViewModel en CreneauxViewModel qui est une classe que j'ai créé qui va contenir une liste de créneaux ainsi que d'autres propriétés de la classe CreneauViewModel.

```
15 références | 0 modification | 0 auteur, 0 modification
public class CreneauxViewModel
{
    2 références | 0 modification | 0 auteur, 0 modification
    public long Id { get; set; }
    7 références | 0 modification | 0 auteur, 0 modification
    public long IdAtelier { get; set; }
    2 références | 0 modification | 0 auteur, 0 modification
    public bool IsArchive { get; set; } = false;
    7 références | 0 modification | 0 auteur, 0 modification
    public bool IsNew { get; set; } = true;
    14 références | 0 modification | 0 auteur, 0 modification
    public List<CreneauViewModel> Creneaux { get; set; }

    4 références | 0 modification | 0 auteur, 0 modification
    public AtelierViewModel AtelierViewModel { get; set; }
}
```

```
@using ReservationCreneau.Core.Reservation.ViewModel
@model CreneauxViewModel
```

J'ai donc dû changer tous les champs html pour qu'ils utilisent les propriétés de cette classe. Exemple de changements effectué :

```
<input asp-for="NombrePlace"
```

```
<input asp-for="@Model.Creneaux[0].NombrePlace"
```

Ensuite, j'ai créé une fonction javascript pour cette duplication qui va être appelée lors du clic sur le bouton de duplication.

```
let creneauCount = 1; // Nombre de créneaux

function duplicateCreneauFields() {

    const creneauFields = document.querySelectorAll('.creneauFields');
    const lastCreneau = creneauFields[creneauFields.length - 1];

    const newCreneau = lastCreneau.cloneNode(true); // Cloner le dernier ensemble de champs
    const inputs = newCreneau.querySelectorAll('input, select'); // Selectionner tout les champs input et select

    // Mettre à jour les attributs name et id pour refléter le nouvel index
    inputs.forEach(input => {
        let label = input.closest('label');

        if(label === null) {
            let nextElement = input.nextElementSibling;

            while (nextElement && nextElement.tagName.toLowerCase() !== 'label') {
                nextElement = nextElement.nextElementSibling; // Passer à l'élément suivant
            }

            if (nextElement) {
                label = nextElement;
            }
        }

        if(label !== null) {
            $(label).attr('for', input.id.replace(/_(\d+)__/, `_${creneauCount}__`));
        }

        input.id = input.id.replace(/_(\d+)__/, `_${creneauCount}__`);
        input.name = input.name.replace(/\[\d+\]/, `[${creneauCount}]`);
    });

    // Masquer le bouton "Dupliquer" dans le dernier ensemble de champs
    const duplicateButton = lastCreneau.querySelector('button[onclick="duplicateCreneauFields()"]');
    if (duplicateButton) {
        duplicateButton.style.display = 'none';
    }

    document.getElementById('creneauContainer').appendChild(newCreneau); // Ajouter les nouveaux champs aux container

    // Ajouter un espace entre les ensembles de champs
    const spacer = document.createElement('div');
    spacer.style.height = '50px';
    document.getElementById('creneauContainer').appendChild(spacer);
```

A noter que cette fonction a été alimentée au fur et à mesure pour résoudre divers problèmes rencontrés que je développerai par la suite.

Dans cette fonction, je vais sélectionner tous les champs qui se trouvent dans la div crenauFields. Ce sont ces champs que je vais dupliquer.

```
<div class="creneauFields">
```

Je vais ensuite cloner ces champs à l'aide de cloneNode().

```
cloneNode(true)
```

Enfin, les champs clonés vont être ajoutés à la suite des premiers dans la div creneauContainer.

```
document.getElementById('creneauContainer').appendChild(newCreneau);
```

Ensuite, j'ai rencontré un problème car tous les champs dupliqués étaient inutilisables car ils avaient tous le même id et name. Pour résoudre ce problème, j'ai créé un compteur qui va

être utilisé pour changer le nom de ces attributs en incrémentant pour chaque nouveau champ dupliqué.

```
const inputs = newCreneau.querySelectorAll('input, select');
```

```
inputs.forEach(input => {
    let label = input.closest('label');

    if(label === null) {
        let nextElement = input.nextElementSibling;

        while (nextElement && nextElement.tagName.toLowerCase() !== 'label') {
            nextElement = nextElement.nextElementSibling; // Passer à l'élément suivant
        }

        if (nextElement) {
            label = nextElement;
        }
    }

    if(label !== null) {
        $(label).attr('for', input.id.replace(/_(\d+)__/, `_${creneauCount}__`));
    }

    input.id = input.id.replace(/_(\d+)__/, `_${creneauCount}__`);
    input.name = input.name.replace(/\[\d+\]/, `[${creneauCount}]`);
});
```

Ici, les deux dernières lignes vont changer l'id et le name de chaque champ. Les conditions et la boucle while vont aller chercher dans la balise label l'attribut for pour également aller le changer.

Ce qui donne :

```
<label class="bpce-input nombre-place" for="Creneaux_1__NombrePlace">
```

```
<input class="bpce-input-field" type="text" pattern="[0-9]*" placeholder="NombrePlace" id="Creneaux_1__NombrePlace" name="Creneaux[1].NombrePlace"
```

Ensuite, pour pouvoir tester mes modifications et afficher la page, j'ai dû modifier toutes les méthodes du contrôleur et du service pour qu'elles reçoivent un objet de la classe CreneauxViewModel vu que le model de la vue avait changé.

```
return View(new CreneauxViewModel() { IdAtelier = idAtelier });
```

```
SaveCreneauAsync(CreneauxViewModel viewModel)
```

Pour les méthodes du service j'ai dû faire une boucle pour pouvoir retourner une liste de créneaux avec l'id de l'atelier auquel il appartient.

```
public async Task CreateCreneauAsync(CreneauxViewModel viewModel)
{
    var creneaux = new List<Creneau>();

    foreach (var creneauViewModel in viewModel.Creneaux)
    {
        var creneau = _mapper.Map<Creneau>(creneauViewModel);
        creneau.IdAtelier = viewModel.IdAtelier;
        creneaux.Add(creneau);
    }

    await _creneauRepository.AddRangeAsync(creneaux);
}
```

Une fois ces modifications apportées, j'ai pu tester la page et la duplication des créneaux. On m'a ensuite demandé d'apporter des modifications pour améliorer la page et les duplications. Je devais :

- Modifier le nom de certains champs qui n'étaient pas corrects (ex. le champ date début qui affichait DateDebut sur la page)
- Retirer certains tooltips qui étaient inutiles
- Faire en sorte que les créneaux dupliqués gardent leurs valeurs dans tous les champs
- Faire disparaître le bouton dupliquer à chaque duplication pour n'apparaître que sur le dernier créneau dupliqué
- Espacer les champs dupliqués entre eux pour plus de lisibilité
- Changer la langue du calendrier du champ date début (qui était en anglais)

Pour les labels incorrects, j'ai juste changé ce qui était affiché sur la page.

```
@Html.DisplayNameFor(model => model.DateDebut)
```

```
>Date début</
```

Pour les tooltips, j'ai simplement retiré les attributs correspondants.

```
data-mdb-toggle="tooltip"
```

```
title="Veuillez renseigner l'heure du début de l'atelier">
```

Pour que les champs gardent leurs valeurs, j'ai modifié ma fonction duplicateCreneauFields. J'ai ajouté une variable qui va contenir les derniers champs dupliqués pour ne garder que leurs valeurs.

```
const lastCreneau = creneauFields[creneauFields.length - 1];
```

```
const newCreneau = lastCreneau.cloneNode(true);
```

Pour faire disparaître le bouton dupliquer et ne le faire que s'afficher que sur le dernier créneau dupliqué, j'ai créé une variable qui va contenir le dernier bouton dupliquer et

l'effacer quand on va cliquer sur le bouton dupliquer existant. Un nouveau bouton va alors être créé lors de la dupliquation des champs.

```
// Masquer le bouton "Dupliquer" dans le dernier ensemble de champs
const duplicateButton = lastCreneau.querySelector('button[onclick="duplicateCreneauFields()"]');
if (duplicateButton) {
    duplicateButton.style.display = 'none';
}
```

Pour espacer les champs, j'ai crée une balise div d'une hauteur de 50px qui va être ajoutée en dessous de chaque ensemble de champ dupliqués.

```
// Ajouter un espace entre les ensembles de champs
const spacer = document.createElement('div');
spacer.style.height = '50px';
document.getElementById('creneauContainer').appendChild(spacer);
```

Enfin, pour changer la date du calendrier pour saisir la date de début du créneau en français, j'ai rencontré un problème. Il existait déjà un élément qui contenait tous les jours et mois en français mais il ne s'appliquait qu'aux champs datepicker or mon champ est un datetime picker. J'ai réussi à implémenter cet élément mais le résultat est que la fenêtre permettant la saisie de l'heure n'apparaissait plus. Pour palier à ce problème, j'ai dû modifier la fonction javascript qui initalise le champ de la date pour y ajouter tous les jours et mois de l'année en français et changer les boutons de validations.

```
function initDatePicker() {
    const datetimepickers = document.querySelectorAll('#datetimepicker-timeOptions');
    datetimepickers.forEach(function (element) {
        new mdb.Datetimepicker(element, {
            timepicker: {
                format24: true,
                okLabel: 'Confirmer',
                clearLabel: 'Effacer',
                cancelLabel: 'Annuler',
            },
            datepicker: {
                title: 'Sélectionner une date',
                monthsFull: ['Janvier', 'Février', 'Mars', 'Avril', 'Mai', 'Juin', 'Juillet', 'Août', 'Septembre', 'Octobre', 'Novembre', 'Décembre'],
                monthsShort: ['Jan', 'Fév', 'Mar', 'Avr', 'Mai', 'Juin', 'Juil', 'Août', 'Sept', 'Oct', 'Nov', 'Déc'],
                weekdaysFull: ['Dimanche', 'Lundi', 'Mardi', 'Mercredi', 'Jeudi', 'Vendredi', 'Samedi'],
                weekdaysShort: ['Dim', 'Lun', 'Mar', 'Mer', 'Jeu', 'Ven', 'Sam'],
                weekdaysNarrow: ['D', 'L', 'M', 'M', 'J', 'V', 'S'],
                okBtnText: 'Ok',
                clearBtnText: 'Effacer',
                cancelBtnText: 'Fermer',
                okBtnLabel: 'Confirmer la sélection',
                clearBtnLabel: 'Effacer la sélection',
                cancelBtnLabel: 'Annuler la sélection',
                startDay: 1,
            }
        });
    });
}
```

Après avoir réalisé ces modifications, j'ai trouvé un problème car un bouton qui était utilisé pour vider les champs envoyait le formulaire. Par exemple, si tout les champs obligatoires pour créer un créneau était saisis et que l'utilisateur cliquais sur ce bouton pour effacer le contenu du champ où il se trouvait, le formulaire était envoyé et le créneau créé.

Ce problème était dû au fait qu'aucun des ces boutons n'avait de type de spécifié, et que par défaut, ils étaient considérés comme des boutons submit.

Enfin, un membre de l'équipe m'a fait part d'un problème lorsqu'il avait utilisé l'application en production. En effet, le bouton pour s'inscrire à un créneau n'apparaissait pas si tout les créneaux appartenant à cet atelier avait des places illimitées. Vu que je travaillais déja sur l'application, il m'a demandé si je pouvais régler ce problème. Je suis allé voir le code de la page concernée et je me suis rendu compte qu'il y avait une condition if qui affichait le bouton d'inscription que si le nombre total de places était strictement supérieur au nombre d'inscriptions valide. Ce qui empêchait le bouton d'apparaître si aucun des crénaux n'avait un nombre de places limitées. Cela posait aussi un problème car si aucune place n' était disponible le bouton ne s'affichait pas non plus. Or, c'était nécéssaire qu'il s'affiche car sur la page d'inscription, il y a un bouton qui permet de signaler que l'on aurait voulu s'inscrire s'il ne reste plus de places ou que la date est dépassée. J'ai donc changé cette condition pour que le bouton ne s'affiche plus uniquement s'il n'éxiste aucun créneau pour l'atelier en question ou si tout les créneaux sont archivés.

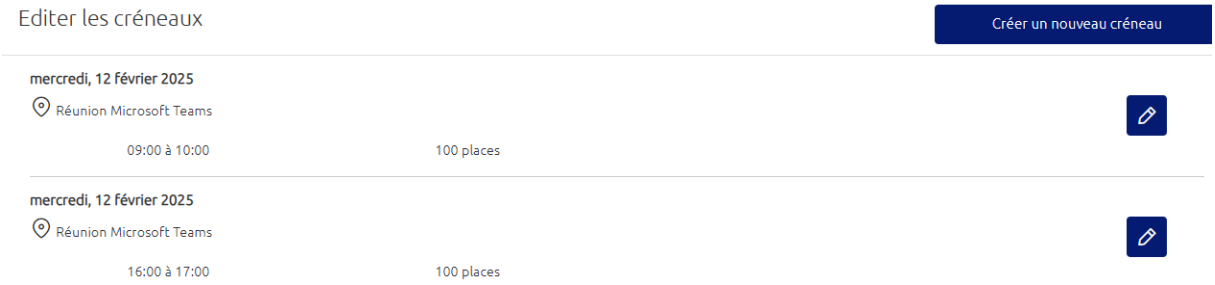
```
if (atelier.Creneaux.Select(c => c.NombrePlace).Sum() > atelier.Creneaux.SelectMany(c => c.Inscriptions).Count(i => i.Valide))
```

```
if (atelier.Creneaux.Any(c => !c.IsArchive))
```

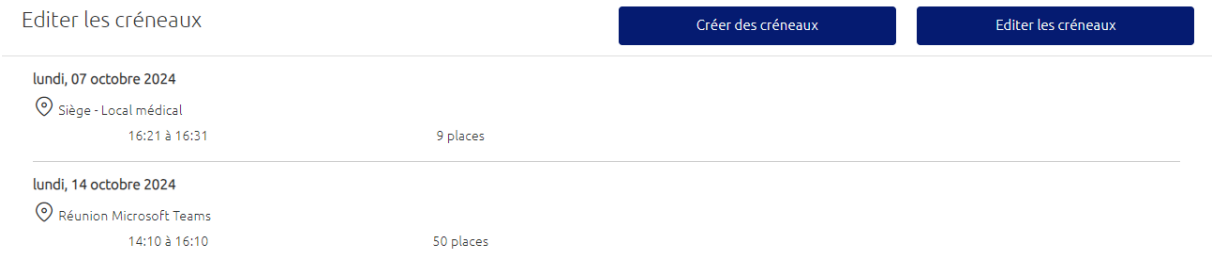
Comme chaque semaine, j'ai participé aux réunions quotidiennes le matin.

Semaine du 10 Février

J'ai continué à travailler sur la duplication de créneaux. J'ai eu trois problèmes majeurs à résoudre pour terminer la duplication. Tout d'abord, j'ai eu un problème avec la page d'édition des créneaux. Après avoir changé l'objet de la méthode du contrôleur utilisée pour l'édition du créneau pour qu'il corresponde au model de la vue, j'ai remarqué que toute modification d'un créneau n'était pas enregistrée dans la base de données et le créneau n'était pas modifié. On m'a dit que c'était parce que j'avais changé l'objet de CreneauViewModel à CreneauxViewModel. L'édition d'un créneau n'étant effectuée que sur un seul créneau à la fois, la liste de créneaux de la classe CreneauxViewModel était vide. On m'a ensuite dit qu'il serait préférable d'avoir la liste des créneaux existants et de pouvoir tous les modifier sur une seule page. Pour cela, j'ai modifié ma vue de création et d'édition de créneaux pour ajouter une condition qui affiche les champs html pour la création d'un créneau quand la propriété isNew est vraie (en l'occurrence quand on clique sur le bouton pour créer des créneaux) et qui va afficher la liste de tous les créneaux existants quand isNew est fausse (quand on clique sur le bouton d'édition des créneaux). Voici les modifications :



Pour donner :




```
onclick="window.location.href='@Url.Action("CreerEditerCreneau", "Evenement", new { idAtelier = Model.Id, isNew = true })'">Créer des créneaux</button>

onclick="window.location.href='@Url.Action("CreerEditerCreneau", "Evenement", new { idAtelier = Model.Id, isNew = false })'">Editer les créneaux</button>
```

Pour l'édition, je vais parcourir la liste des créneaux et afficher chaque ensemble de champs pour chaque créneau différent.

```
<div id="creneauContainer">
    @for (int i = 0; i < Model.Creneaux.Count; i++)
    {
        <input asp-for="@Model.Creneaux[i].Id" type="hidden" />
        <input asp-for="@Model.Creneaux[i].IdAtelier" type="hidden" />
        <div class="creneauFields bpce-card bpce-card-padding-sm mb-5">
            <div class="row">
                <div class="col-2">
                    <label asp-for="@Model.Creneaux[i].NombrePlace" class="bpce-input nombre-place">
                        <div class="bpce-input-container" data-mdb-toggle="tooltip" data-mdb-html="true" title="Inscrivez
                            <input asp-for="@Model.Creneaux[i].NombrePlace" class="bpce-input-field" type="text" pattern=
                            <span class="bpce-input-label">Nombre de places</span>
                            <button type="button" class="btn-erase bpce-button bpce-button-icon-only bpce-button-no-borde
                                <span class="bpce-icon-font erase" aria-hidden="true"></span>
                            </button>
                        </div>
                    </label>
                </div>
                <div class="col-3">
                    <div class="form-outline bpce-input bpvf-datepicker" id="datetimepicker-timeOptions" data-mdb-input-i
                        <input asp-for="@Model.Creneaux[i].DateDebut" data-mdb-toggle="datetimepickertimeOptions" aria-la
                        <label asp-for="@Model.Creneaux[i].DateDebut" class="bpce-input-label bpce-input bpvf-datepicker-
                        <button class="datetimepicker-toggle-button" type="button" data-mdb-toggle="datetimepicker">
                            <i class="bpce-icon-font bpce-calendar"></i>
                        </button>
                    </div>
                </div>
```

J'ai ensuite modifié la méthode de sauvegarde des créneaux pour appeler la bonne méthode selon si on veut créer ou modifier des créneaux.

```
[HttpPost]
0 références | William GRIMAUD, Il y a 40 minutes | 1 auteur, 1 modification
public async Task<IActionResult> SaveCreneauAsync(CreneauxViewModel viewModel)
{
    if (viewModel.IsNew)
    {
        await _evenementService.CreateCreneauxAsync(viewModel);
    }
    else
    {
        await _evenementService.UpdateCreneauxAsync(viewModel);
    }

    return Json(viewModel.IdAtelier);
}
```

J'ai également modifié la méthode UpdateCrenauxAsync pour mapper la liste des créneaux pour qu'elle puisse fonctionner avec.

```
2 références | William GRIMAUD, Il y a 47 minutes | 1 auteur, 1 modification
public async Task UpdateCreneauxAsync(CreneauxViewModel viewModel)
    => await _creneauRepository.UpdateRangeAsync(_mapper.Map<List<Creneau>>(viewModel.Creneaux));
```

Une fois ces modifications apportées, la page d'édition fonctionnait et l'enregistrement des créneaux modifié était effectué.

Ensuite, j'ai eu un deuxième problème car certaines fonction javascript de la page utilisait les id des champs. Or, ces id étant incorrects après avoir ajouté l'incrémentation des id et name des champs je ne savais pas comment modifier les fonctions les utilisant. Après avoir effectué des recherches et demandé de l'aide, j'ai ajouté une classe sur la div du champ utilisé dans la fonction. Grâce à ça j'ai modifié la fonction qui permet d'afficher le champ texte du lien de réunion teams que quand le lieu sélectionné dans la liste déroulante est réunion Microsoft teams et le cacher si ce n'est pas le cas.

```
// Affiche le lien que si Teams est sélectionné
function toggleLienTeams(selectElement) {
    let creneauFields = selectElement.closest('.creneauFields');

    let champLien = creneauFields.find('.champLien');

    if (selectElement.val() == 5) {
        champLien.css('display', 'flex');
    } else {
        champLien.css('display', 'none');
    }
}
```

J'ai également modifié la fonction qui va vérifier si le lien teams est bien valide.

```
// Vérifie si le lien est valide
function lienValid(inputElement) {
    if (inputElement.attr('placeholder') === "Lien") {
        const lienValue = inputElement.val();
        var pattern = /^(ftp|http|https):\/\/[^ "]+$/;
        //var pattern = /^https:\/\/teams\.microsoft\.com\/[^ "]+$/;

        if (pattern.test(lienValue)) {
            inputElement.closest('.champLien').removeClass("bpce-input-error");
            inputElement.css('color', '');
            inputElement.closest('.champLien').find('.btn-effacer').css('color', '');
            return true;
        } else {
            alert('Le lien n\'est pas correct');
            inputElement.closest('.champLien').addClass("bpce-input-error");
            inputElement.css('color', 'var(--color-error)');
            inputElement.closest('.champLien').find('.btn-effacer').css('color', 'var(--color-error)');
            return false;
        }
    }
}
```

Ici, on va aller chercher le champ avec le placeholder lien et vérifier avec une expression régulière si le lien est valide et sinon envoyer une alerte.

Enfin, j'ai modifié la fonction appelée à l'envoi du formulaire pour empêcher l'envoi si le lien est incorrect.

```
const formData = new FormData(oFormElement);
const creneauxFields = oFormElement.querySelectorAll('[name*="Creneaux"]');

for (let field of creneauxFields) {
    if (field.name.includes('IdLieu') && field.value === "5") {
        const lienElement = oFormElement.querySelector(`[name="${field.name.replace('IdLieu', 'Lien')}"]`);
        if (lienElement && !lienValid($(lienElement))) {
            return false;
        }
    }
}

const response = await fetch(oFormElement.action, {
    method: 'POST',
    body: formData,
    headers: new Headers()
});
if (response.ok) {
    window.location.href = '@Url.Action("Edit", "Atelier")' + '/' + await response.json();
}
else {
    changeToasterState89C3("Une erreur s'est produite", 'bpce-toaster-loader-error', true, false);
}
```

Dans le document ready, je vais mettre la valeur du champ durée à 0 par défaut car elle était à 12 auparavant. C'est également ici que je vais appeler les fonctions modifiées précédemment.

```
if (document.getElementById("IsNew").value == "True") {
    $('input[name*="Duree"]').each(function() {
        $(this).val("00:00");
    });
} else {
    $('input[name*="Duree"]').each(function() {
        $(this).val($(this).val());
    });
}

// initTextAreaWatchOutWords();

$('.select-lieu').each(function() {
    toggleLienTeams($(this));
});

$(document).on('change', '.select-lieu', function() {
    toggleLienTeams($(this));
});

$(document).on("blur", ".bpce-input-field[type='text']", function () {
    lienValid($(this));
});
```

La fonction toggleLienTeams qui affiche le champ du lien est appelée deux fois : une fois quand la valeur de la liste déroulante change et une autre fois pour la page d'édition pour afficher le champ si le lieu est déjà défini comme une réunion teams. La fonction lienValid est appelée lorsque l'utilisateur n'est pas sur le champ du lien (si l'utilisateur clique sur le champ du lien et clique ailleurs sans avoir rentré un lien valide, l'alerte se déclenchera).

Après avoir réglé le problème de l'affichage du lien teams, il me restait un dernier problème. Lors de la duplication de champs si l'utilisateur ne clique pas sur le dernier champ de la date début dupliqué, les autres champs ne fonctionneront pas car toutes les fenêtres affichant le calendrier pour saisir la date s'ouvriront en même temps ce qui rendra la saisie impossible (ce problème étant le même pour la fenêtre qui s'affiche pour saisir la durée). Pour régler ce problème, j'ai modifié les fonctions qui initialisent les champs de la date et de la durée pour qu'elles reçoivent en paramètre l'élément sur le lequel elles vont être initialisées.

```
function initDatePicker(container) {
    const datetimepickers = container.querySelectorAll('#datetimepicker-timeOptions');
```

```
function initTimePicker(container) {
    const timepickers = container.querySelectorAll('.timepicker-translated');
```

Dans ma fonction qui va dupliquer les créneaux, je vais passer en paramètre uniquement les derniers champs dupliqués pour ne pas initialiser la date et la durée sur tous les champs à chaque duplication.

```
initDatePicker(newCreneau);
initTimePicker(newCreneau);
```

Et je vais faire de même dans le document ready pour les initialiser sur les premiers champs.

```
$(document).ready(function () {

    initDatePicker(document);
    initTimePicker(document);
```

Le problème était ensuite réglé et tous les champs de date et de durée fonctionnaient correctement.

Pour terminer cette mission, le membre de l'équipe qui m'a donné cette mission a réalisé des tests pour s'assurer que tout fonctionnait correctement. Après avoir effectué ces tests, plusieurs choses était à changer. D'abord, le bouton pour éditer les créneaux s'affichait même s'il n'y avait pas de créneaux sur cet atelier. Ensuite, lors de la duplication des champs, la liste déroulant de gardait pas ça valeur sélectionnée alors que cela doit être le cas. Enfin, également lors de la duplication, les champs dupliqués sont trop collés et il faut qu'ils soient plus lisibles et compréhensibles.

J'ai commencé par n'afficher le bouton pour éditer les créneaux que quand il y a déjà des créneaux existants en ajoutant un condition if.

```
@if (Model.CreneauxViewModel.Any())
{
    <div class="col-6">
        <h4>Editer les créneaux</h4>
    </div>
    <div class="col-md-3 bpce-button-group">
        <button type="button" class="bpce-button bpce-button-flex bpc
    </div>
    <div class="col-md-3 bpce-button-group">
        <button type="button" class="bpce-button bpce-button-flex bpc
    </div>
}
else
{
    <div class="col-9">
        <h4>Editer les créneaux</h4>
    </div>
    <div class="col-md-3 bpce-button-group">
        <button type="button" class="bpce-button bpce-button-flex bpc
    </div>
}
```

Ensuite, pour garder la valeur de la liste déroulante lors de la duplication, j'ai récupéré sa valeur précédente et j'ai l'ai affectée à la liste déroulante dupliquée.

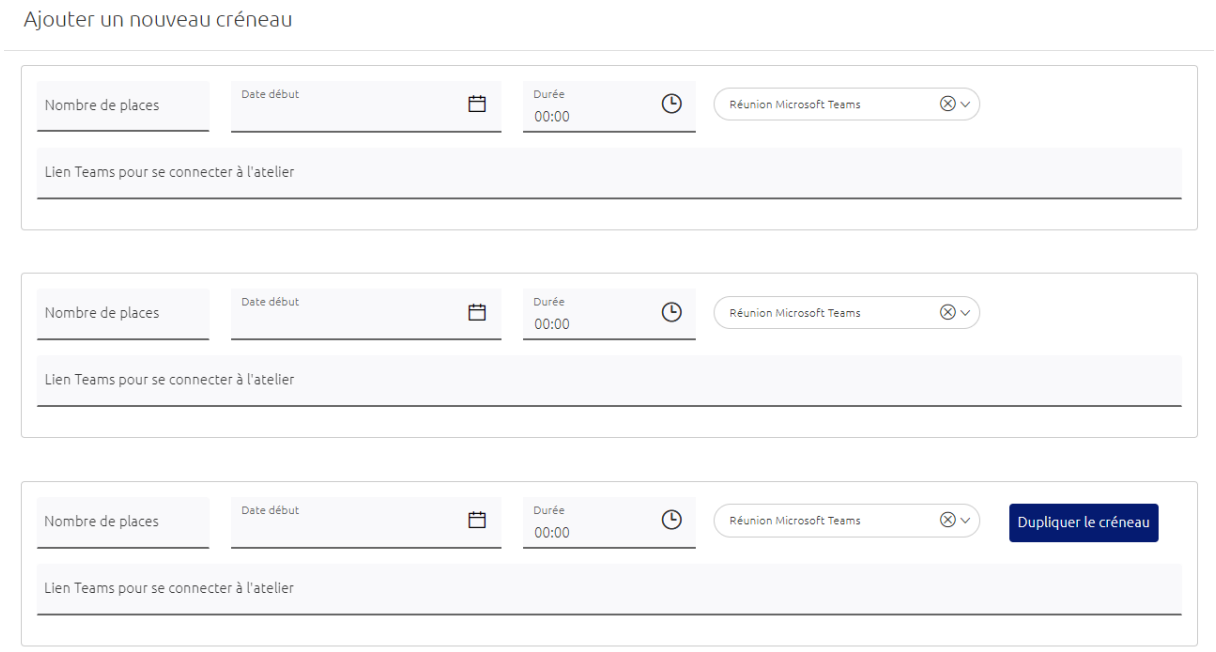
```
const selectLieu = lastCreneau.querySelector('select');
```

```
//Garder la valeur du select après chaque duplication
if(input.type == 'select-one') {
    input.value = selectLieu.value;
}
```

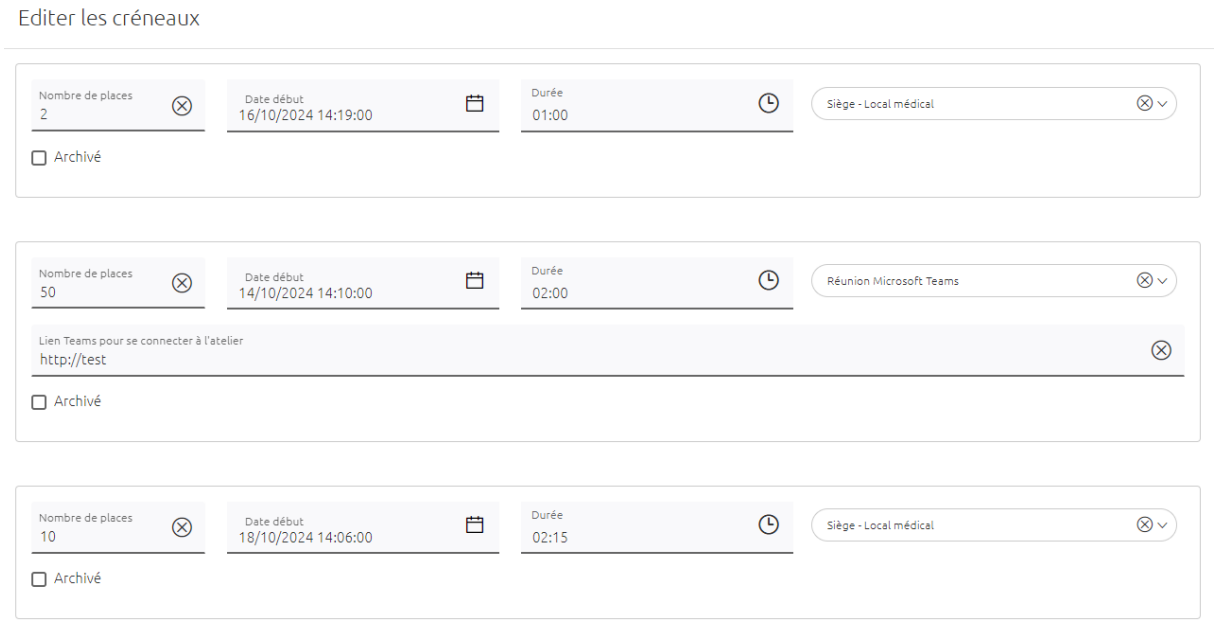
Enfin, pour rendre les champs plus lisibles j'ai ajouté un contour pour délimiter chaque champ appartenant à un créneau avec des classes css déjà existantes.

```
<div class="creneauFields bpce-card bpce-card-padding-sm mb-5">
```

Résultat sur la page de création :



Résultat sur la page d'édition :



Après avoir réalisé ces modifications, j'ai poussé mon code et un membre de l'équipe a revu mon code. J'ai eu une méthode d'un service à modifier pour qu'elle soit écrite de la même manière que d'autres méthodes similaires.

```
-     public async Task<Collaborateur> GetCollabByIdsCollabAsync(int idCollab)
-     {
-         return await _collaborateurRepository.GetByIdAsync(idCollab);
-     }
```

```
10
11  +     public async Task<Collaborateur> GetCollabByIdCollabAsync(int idCollab)
12  +         => await _collaborateurRepository.GetByIdAsync(idCollab);
```

Une fois fait, la branche sur laquelle j'ai travaillé a été fusionnée par ce même membre. Avant le déploiement en production, on m'a demandé des réaliser des derniers tests pour s'assurer que l'application marche correctement et qu'il n'y ait pas de crashs.

Après avoir réalisés ces tests sans encombre, l'application a été déployée en production et les migrations ont été réalisées sur la base de données de production.

Bilan de stage

Mesure de la réussite du projet

J'ai réussi toutes les tâches qui m'ont été données pendant ce stage de 6 semaines.

J'ai ajouté divers filtres sur un application qui répertorie touts les projets du service. J'ai également, sur une application de création d'ateliers et de créneaux ainsi que de réservation, ajouté la possibilité d'exporter sur Excel les statistiques de réservation et la possibilité de dupliquer des créneaux lors de leur création. Enfin, j'ai corrigé un problème existant qui avait été remonté par un membre de l'équipe lors de l'utilisation d'application et auquel on m'a assigné la résolution.

Tout ce qui a été prévu a été réalisé, on m'a donné une nouvelle tâche dès que j'avais fini la tâche précédente qui m'avait été donnée.

Ce que m'a apporté le stage

Ce stage m'a permis d'approfondir ce que j'avais appris sur le Framework ASP .NET CORE, la programmation web orientée objet en C# et les requêtes sur une base de données en utilisant l'ORM EntityFramework. J'ai également utilisé JQuery et Bootstrap ce qui m'a aussi permis consolider mes connaissances. J'ai aussi appris à réaliser des migrations pour créer des colonnes dans une base de données en utilisant une démarche code first.

J'ai continué à utiliser Azure DevOps comme gestionnaire de code source en récupérant le code existant, en réalisant des validations et des requêtes de tirage, en créant des branches et faisant revoir mon code par un membre de l'équipe.

Enfin, j'ai appris à m'intégrer et travailler au sein d'une équipe en participant à des réunions quotidiennes et en sollicitant des membres de l'équipe en cas de besoin. J'ai également appris à travailler dans un cadre professionnel au sein d'une entreprise et dans un cadre réglementé en étant vigilant à toute tentative de phishing et aux données sensibles.

Conclusion

Je suis satisfait de ce stage et j'ai appris de nombreuses choses pendant celui-ci. J'ai pu mettre en pratique ce que j'ai appris pendant l'année at approfondir ce que j'ai appris l'année dernière. J'ai pu acquérir de nouvelles compétences et consolider mes acquis. Enfin, j'ai pu davantage découvrir le travail en entreprise et en équipe.

Page de remerciements

Je remercie le service Développement et Sécurité Informatique de m'avoir accueilli une seconde fois et de m'avoir accompagné pendant ces 6 semaines de stage.

Je remercie Thomas DORVAL de m'avoir accepté et accueilli une deuxième fois en stage dans son service et de m'avoir accompagné durant mon stage.

Je remercie Céline GIBAUD et Valérie BETTIN de m'avoir accepté au sein de la direction Systèmes d'Information.

Enfin, je remercie la Banque Populaire Val de France de m'avoir ouvert une deuxième fois ses portes pour mon stage de seconde année.