Améliorations du Système de Contrats NORM

Résumé des modifications

Ce document récapitule les 3 améliorations apportées au système de contrats NORM pour corriger les problèmes identifiés.

---

1. ✅ Payroll User dans les Participants

Problème

Le payroll user était stocké dans le champ `payrollUserId` du contrat, ce qui n'était pas cohérent avec le reste du système où les participants sont gérés via la table `ContractParticipant`.

Solution

Le payroll user est maintenant ajouté comme participant avec le rôle `"payroll"` dans la table `ContractParticipant`.

Modifications apportées

Backend - `server/api/routers/simpleContract.ts`

Dans `createNormContract` (après ligne 1513) :

```
// Participant 4 (optionnel): Payroll User (si salaryType = payroll ou payroll_we_pay)
if ((salaryType === "payroll" || salaryType === "payroll_we_pay") && payrollUserId) {
  await createMinimalParticipant(ctx.prisma, {
    contractId: contract.id,
    userId: payrollUserId,
    role: "payroll",
    isPrimary: false,
  });
}
```

Dans `updateNormContract` (après ligne 1674) :

```
// 5b. Gérer la mise à jour du participant payroll si nécessaire
if (updateData.payrollUserId !== undefined) {
  // Supprimer l'ancien participant payroll
  await ctx.prisma.contractParticipant.deleteMany({
    where: {
      contractId,
      role: "payroll",
    },
  });

  // Créer un nouveau participant payroll si payrollUserId est fourni
  if (updateData.payrollUserId && (updated.salaryType === "payroll" || up-
dated.salaryType === "payroll_we_pay")) {
    await createMinimalParticipant(ctx.prisma, {
      contractId,
      userId: updateData.payrollUserId,
      role: "payroll",
      isPrimary: false,
    });
  }
}
```

Impact

- Le payroll user est maintenant cohérent avec les autres participants (Company Tenant, Agency, Contractor)
- Les queries existantes (`getSimpleContractById` , `listSimpleContracts`) incluent déjà les participants, donc aucune modification supplémentaire n'est nécessaire
- Le champ `payrollUserId` reste dans le contrat pour compatibilité, mais le système utilise maintenant les participants

---

2. ✅ Affichage et Téléchargement du PDF

Problème

- Le composant `ContractDocumentViewer` affichait uniquement un placeholder
- Le PDF n'était pas visible
- Le téléchargement n'était pas implémenté

Solution

- Utilisation de l'endpoint tRPC existant `document.getSignedUrl` pour récupérer l'URL signée S3
- Affichage du PDF dans un iframe
- Implémentation du téléchargement réel avec création d'un blob

Modifications apportées

Frontend - `components/contracts/simple/ContractDocumentViewer.tsx`

Ajout des imports :

```
import { useState, useEffect } from "react";
import { api } from "@/lib/trpc";
```

Récupération de l'URL signée :

```
const [pdfUrl, setPdfUrl] = useState<string | null>(null);

// Récupérer l'URL signée du document
const { data: signedUrlData, isLoading: isLoadingUrl } = api.document.getSigned-
Url.useQuery(
  { documentId: document.id },
  {
    enabled: !!document.id,
    staleTime: 1000 * 60 * 50, // 50 minutes (les URLs S3 expirent après 1h)
  }
);

useEffect(() => {
  if (signedUrlData?.url) {
    setPdfUrl(signedUrlData.url);
  }
}, [signedUrlData]);
```

Affichage du PDF dans un iframe :

```
{isLoadingUrl ? (
  <div className="h-[600px] flex items-center justify-center bg-muted/20">
    <Loader2 className="h-8 w-8 animate-spin text-muted-foreground" />
    <p className="text-sm text-muted-foreground">Chargement du document...</p>
  </div>
) : pdfUrl ? (
  <iframe
    src={`${pdfUrl}#toolbar=0&navpanes=0`}
    className="w-full h-[600px] border-0"
    title={document.fileName}
  />
) : (
  <div className="h-[600px] flex flex-col items-center justify-center gap-3 bg-muted/
20">
    <FileText className="h-16 w-16 text-muted-foreground" />
    <p className="text-sm text-muted-foreground text-center">
      Impossible de charger le document
    </p>
  </div>
)}
```

Téléchargement fonctionnel :

```
const handleDownload = async () => {
  setIsDownloading(true);
  try {
    if (!pdfUrl) {
      toast.error("URL du document non disponible");
      return;
    }

    // Télécharger le fichier
    const response = await fetch(pdfUrl);
    const blob = await response.blob();
    const url = window.URL.createObjectURL(blob);
    const link = window.document.createElement("a");
    link.href = url;
    link.download = document.fileName;
    link.click();
    window.URL.revokeObjectURL(url);

    toast.success("Document téléchargé avec succès");
  } catch (error) {
    console.error("[ContractDocumentViewer] Download error:", error);
    toast.error("Erreur lors du téléchargement");
  } finally {
    setIsDownloading(false);
  }
};
```

Bouton pour ouvrir dans un nouvel onglet :

```
{pdfUrl && (
  <Button
    variant="outline"
    size="sm"
    onClick={() => window.open(pdfUrl, "_blank")}
  >
    <ExternalLink className="mr-2 h-4 w-4" />
    Ouvrir dans un nouvel onglet
  </Button>
)}
```

Impact

- Les utilisateurs peuvent maintenant **voir le PDF directement** dans l'interface
- Le **téléchargement fonctionne** correctement
- L'URL signée S3 est mise en cache pendant 50 minutes pour éviter les requêtes inutiles
- L'affichage gère les états de chargement, d'erreur et de succès

---

3. ✅ Affichage des Parties du Contrat

Problème

- Le composant `NormContractView` utilisait des champs inexistants : `contract.companyTenant`, `contract.agency`, `contract.contractor`, `contract.payrollUser`
- Ces champs n'existent pas dans le type `Contract`
- Les participants n'étaient pas affichés correctement

Solution

- Utilisation de `contract.participants` pour récupérer les parties
- Filtrage par rôle : `"tenant"`, `"agency"`, `"contractor"`, `"payroll"`
- Affichage conditionnel du payroll user (uniquement si présent)

Modifications apportées

Frontend - `components/contracts/simple/NormContractView.tsx`

Mise à jour de l'interface :

```
interface NormContractViewProps {
  contract: {
    // ...
    // Participants (pour récupérer les parties)
    participants?: Array<{
      id: string;
      role: string;
      user?: { id: string; name: string | null; email: string } | null;
      company?: { id: string; name: string } | null;
    }>;
    // Suppression des champs : companyTenant, agency, contractor, payrollUser
  };
}
```

Extraction des participants par rôle :

```
// Récupérer les participants par rôle
const participants = contract.participants || [];
const companyTenant = participants.find((p) => p.role === "tenant");
const agency = participants.find((p) => p.role === "agency");
const contractor = participants.find((p) => p.role === "contractor");
const payrollUser = participants.find((p) => p.role === "payroll");
```

Affichage des parties :

```
<Card>
  <CardHeader>
    <CardTitle>Parties du contrat</CardTitle>
  </CardHeader>
  <CardContent className="space-y-3">
    {/* Company Tenant */}
    <div className="flex items-start gap-3 p-3 rounded-lg border">
      <Building2 className="h-5 w-5 text-muted-foreground flex-shrink-0 mt-0.5" />
      <div className="flex-1">
        <p className="text-sm text-muted-foreground">Company Tenant</p>
        <p className="font-medium">
          {companyTenant?.company?.name || companyTenant?.user?.name ||
companyTenant?.user?.email || "-"}
        </p>
      </div>
    </div>

    {/* Agency */}
    <div className="flex items-start gap-3 p-3 rounded-lg border">
      <Building2 className="h-5 w-5 text-muted-foreground flex-shrink-0 mt-0.5" />
      <div className="flex-1">
        <p className="text-sm text-muted-foreground">Agency</p>
        <p className="font-medium">
          {agency?.company?.name || agency?.user?.name || agency?.user?.email || "-"}
        </p>
      </div>
    </div>

    {/* Contractor */}
    <div className="flex items-start gap-3 p-3 rounded-lg border">
      <User className="h-5 w-5 text-muted-foreground flex-shrink-0 mt-0.5" />
      <div className="flex-1">
        <p className="text-sm text-muted-foreground">Contractor</p>
        <p className="font-medium">
          {contractor?.user?.name || contractor?.user?.email || contractor?.company?.n
ame || "-"}
        </p>
      </div>
    </div>

    {/* Payroll User (conditionnel) */}
    {payrollUser && (
      <div className="flex items-start gap-3 p-3 rounded-lg border">
        <User className="h-5 w-5 text-muted-foreground flex-shrink-0 mt-0.5" />
        <div className="flex-1">
          <p className="text-sm text-muted-foreground">Payroll User</p>
          <p className="font-medium">
            {payrollUser?.user?.name || payrollUser?.user?.email || "-"}
          </p>
        </div>
      </div>
    )}
  </CardContent>
</Card>
```

Mise à jour de la section “Type de salaire et paiement” :

```
{(contract.salaryType === "payroll" || contract.salaryType === "payroll_we_pay") &&
payrollUser && (
  <div className="pt-3 border-t">
    <p className="text-sm text-muted-foreground flex items-center gap-2">
      <User className="h-4 w-4" />
      Utilisateur Payroll
    </p>
    <p className="font-medium mt-1">
      {payrollUser.user?.name || payrollUser.user?.email || "-"}
    </p>
    {contract.salaryType === "payroll_we_pay" && (
      <p className="text-xs text-muted-foreground mt-1">(Géré par le système)</p>
    )}
  </div>
)}
```

Impact

- Les parties du contrat sont maintenant **correctement affichées**
- Le système utilise la table `ContractParticipant` comme source unique de vérité
- Le payroll user s'affiche uniquement s'il est présent (pour les types "payroll" et "payroll_we_pay")
- Gestion des cas où le participant est un `user` ou une `company`
- Affichage de "-" pour les champs non renseignés

Fichiers modifiés

Backend

1. **`server/api/routers/simpleContract.ts`**
   - Ajout de la création du participant payroll dans `createNormContract`
   - Ajout de la gestion de la mise à jour du participant payroll dans `updateNormContract`

Frontend

1. **`components/contracts/simple/ContractDocumentViewer.tsx`**
   - Ajout de la récupération de l'URL signée S3 via tRPC
   - Affichage du PDF dans un iframe
   - Implémentation du téléchargement réel

2. **`components/contracts/simple/NormContractView.tsx`**
   - Modification de l'interface pour utiliser `participants` au lieu des champs directs
   - Extraction des participants par rôle
   - Affichage conditionnel des parties du contrat
   - Mise à jour de la section payroll user

Tests recommandés

1. Création d'un contrat NORM

- [ ] Créer un contrat NORM avec `salaryType = "payroll"`
- [ ] Vérifier que le payroll user est bien ajouté comme participant

- [ ] Vérifier que le rôle est bien `"payroll"`

2. Mise à jour d'un contrat NORM

- [ ] Modifier le `payrollUserId` d'un contrat existant
- [ ] Vérifier que l'ancien participant payroll est supprimé
- [ ] Vérifier que le nouveau participant payroll est créé

3. Affichage du PDF

- [ ] Ouvrir un contrat NORM avec un document
- [ ] Vérifier que le PDF s'affiche dans l'iframe
- [ ] Tester le téléchargement du PDF
- [ ] Tester l'ouverture dans un nouvel onglet

4. Affichage des parties

- [ ] Ouvrir un contrat NORM
- [ ] Vérifier que Company Tenant, Agency, Contractor s'affichent correctement
- [ ] Vérifier que Payroll User s'affiche uniquement si présent
- [ ] Vérifier l'affichage du payroll user dans la section "Type de salaire et paiement"

Notes importantes

Compatibilité

- Le champ `payrollUserId` est conservé dans le schéma Contract pour compatibilité ascendante
- Les queries existantes (`getSimpleContractById` , `listSimpleContracts`) incluent déjà les participants
- Aucune migration de données n'est nécessaire pour les contrats existants

Performance

- L'URL signée S3 est mise en cache pendant 50 minutes pour réduire les requêtes
- Les participants sont chargés en une seule requête avec les includes optimisés

Sécurité

- Les URLs S3 sont signées et expirent après 1 heure
- Les permissions tRPC sont vérifiées pour l'accès aux documents
- Les participants sont filtrés par `isActive = true`

Prochaines étapes recommandées

1. **Migration optionnelle** : Créer un script de migration pour convertir les contrats NORM existants qui ont un `payrollUserId` en participants
2. **Tests E2E** : Ajouter des tests end-to-end pour vérifier le workflow complet
3. **Documentation utilisateur** : Mettre à jour la documentation pour refléter les changements

---

Date de mise en œuvre : 30 novembre 2025
Développeur : DeepAgent
Statut : ✅ Compilé et prêt pour les tests