Expense Structure Implementation

Vue d'ensemble

Cette implémentation corrige la structure des expenses dans le système payroll-saas en créant une table `Expense` séparée et en liant correctement les expenses aux timesheets et aux invoices.

Changements apportés

1. Schéma Prisma (prisma/schema.prisma)

Modèle Expense

- ✅ Ajout du champ `timesheetId` pour lier les expenses aux timesheets
- ✅ Ajout du champ `documentId` pour lier les expenses aux documents (receipts)
- ✅ Ajout de la relation `timesheet` vers le modèle Timesheet
- ✅ Ajout de la relation `document` vers le modèle Document
- ✅ Ajout des index appropriés pour `timesheetId` et `documentId`

Modèle Timesheet

- ✅ Ajout de la relation `expenses` vers le modèle Expense

Modèle Document

- ✅ Ajout de la relation `expenses` vers le modèle Expense

2. Backend (server/api/routers/timesheet.ts)

Fonction `createRange`

- ✅ Modifié pour créer des entrées dans la table `Expense` au lieu de `TimesheetDocument`
- ✅ Chaque expense est maintenant une entité séparée avec :
- `title` (catégorie de l'expense)
- `description`
- `amount`
- `currency`
- `category`
- `receiptUrl` (optionnel)
- `timesheetId` (lien vers le timesheet)
- `contractId` (lien vers le contrat)
- `submittedBy` (userId)

Fonction `getById`

- ✅ Modifié pour inclure les `expenses` dans la requête Prisma
- ✅ Retourne maintenant les expenses depuis la table Expense

Fonction `sendToAgency`

- ✅ Modifié pour calculer `totalExpenses` depuis la table Expense
- ✅ Crée des `InvoiceLine` pour chaque expense individuelle
- ✅ Chaque expense apparaît comme une ligne séparée dans l'invoice avec :

- Description : “Expense: {title} - {description}”
- Quantity : 1
- Unit Price : montant de l’expense
- Amount : montant de l’expense
- ✅ Met à jour les notes de l’invoice pour lister toutes les expenses

3. Migration Prisma

Une migration Prisma doit être créée pour ajouter les nouveaux champs :

```
npx prisma migrate dev --name add-expense-timesheet-link
```

Cette migration ajoutera :
- Colonne `timesheetId` à la table `expenses`
- Colonne `documentId` à la table `expenses`
- Index sur `timesheetId` et `documentId`
- Contraintes de clé étrangère appropriées

Structure des données

Avant (❌ Incorrect)

```
Timesheet
  ├── TimesheetEntry (heures travaillées)
  ├── TimesheetDocument (fichiers d'expenses)
  └── totalExpenses (montant total)

Invoice
  └── InvoiceLineItem (seulement les heures)
```

Après (✅ Correct)

```
Timesheet
  ├── TimesheetEntry (heures travaillées)
  ├── Expense (expenses individuelles)
  │   ├── amount
  │   ├── description
  │   ├── category
  │   └── documentId → Document (receipt)
  └── totalExpenses (calculé depuis Expense[])

Invoice
  └── InvoiceLineItem
      ├── Lignes pour les heures travaillées
      └── Lignes pour chaque Expense
```

Flux de données

1. **Création d’un timesheet** :
   - L’utilisateur crée un timesheet avec des heures et des expenses
   - Les expenses sont créées dans la table `Expense` avec `timesheetId`
   - Chaque expense peut avoir un `documentId` lié à un receipt

2. **Approbation du timesheet** :
   - Le timesheet passe par le workflow d'approbation
   - Les expenses sont approuvées en même temps que le timesheet

3. **Envoi à l'agence (Send to Agency)** :
   - Une invoice est créée avec des `InvoiceLine` pour :
     - Chaque jour travaillé (heures × taux)
     - Chaque expense individuelle
     - Le calcul de l'invoice inclut :
     - Base amount (heures uniquement)
     - Total expenses (somme des expenses)
     - Margin (appliquée sur base + expenses)
     - Total amount (base + expenses + margin)

4. **Affichage de l'invoice** :
   - Les line items montrent clairement :
     - Les jours travaillés avec les heures
     - Les expenses avec leur description et montant
     - Le calcul est transparent et détaillé

Avantages de cette structure

1. **Séparation des préoccupations** :
   - Les expenses sont des entités séparées, pas juste des documents
   - Chaque expense a ses propres métadonnées (montant, catégorie, etc.)

2. **Traçabilité** :
   - Chaque expense est liée à un timesheet spécifique
   - Chaque expense peut avoir un document (receipt) lié
   - L'historique des expenses est clair

3. **Flexibilité** :
   - Les expenses peuvent être approuvées/rejetées individuellement
   - Les expenses peuvent être modifiées avant l'envoi de l'invoice
   - Les expenses peuvent être liées à des workflows d'approbation

4. **Transparence dans les invoices** :
   - Chaque expense apparaît comme une ligne séparée
   - Le client peut voir exactement ce qui est facturé
   - Les calculs sont clairs et vérifiables

Prochaines étapes

Frontend (à implémenter)

1. **Affichage du timesheet** :
   - Modifier les composants pour afficher les expenses depuis la table Expense
   - Afficher chaque expense avec son montant, catégorie, et description
   - Permettre l'ajout/modification/suppression d'expenses

2. **Affichage de l'invoice** :
   - Modifier `InvoiceDetailModal.tsx` pour grouper les line items :
     - Section "Work Hours" pour les heures travaillées
     - Section "Expenses" pour les expenses
     - Afficher le calcul détaillé :
     - Subtotal (Hours)
     - Expenses
     - Base Amount
     - Margin
     - Total Amount
3. **Formulaire de création de timesheet** :
   - Ajouter une section pour les expenses
   - Permettre l'upload de receipts pour chaque expense
   - Valider que chaque expense a un montant et une catégorie

Migration des données existantes

Si des timesheets existants ont des `TimesheetDocument` avec des expenses, une migration de données sera nécessaire :

```
// Script de migration (à exécuter une fois)
const timesheets = await prisma.timesheet.findMany({
  include: { documents: true }
});

for (const ts of timesheets) {
  if (ts.documents.length > 0 && ts.totalExpenses > 0) {
    // Créer une expense pour chaque document
    for (const doc of ts.documents) {
      await prisma.expense.create({
        data: {
          tenantId: ts.tenantId,
          timesheetId: ts.id,
          contractId: ts.contractId,
          submittedBy: ts.submittedBy,
          title: "Migrated Expense",
          description: doc.description,
          amount: ts.totalExpenses / ts.documents.length, // Répartir équitablement
          currency: ts.currency || "USD",
          category: "other",
          documentId: doc.id,
          expenseDate: ts.startDate,
          status: ts.status,
        }
      });
    }
  }
}
```

Tests recommandés

1. **Test de création de timesheet avec expenses** :
   - Créer un timesheet avec 2-3 expenses

- Vérifier que les expenses sont créées dans la table Expense
- Vérifier que `totalExpenses` est calculé correctement

2. **Test de sendToAgency** :
   - Approuver un timesheet avec expenses
   - Envoyer à l'agence
   - Vérifier que l'invoice contient des line items pour chaque expense
   - Vérifier que le calcul est correct

3. **Test d'affichage** :
   - Afficher un timesheet avec expenses
   - Vérifier que les expenses sont affichées correctement
   - Afficher l'invoice générée
   - Vérifier que les line items sont groupés correctement

Conclusion

Cette implémentation corrige la structure des expenses en créant une table séparée et en liant correctement les expenses aux timesheets et aux invoices. Les expenses sont maintenant des entités de première classe avec leurs propres métadonnées et relations, ce qui permet une meilleure traçabilité, flexibilité et transparence dans le système.