Correctifs nécessaires pour aligner l'API de mission

Problèmes identifiés

Après analyse du contrôleur backend (`MissionControleur.java`) et du service frontend (`MissionService.js`), plusieurs incohérences ont été identifiées qui pourraient causer des problèmes lors de la mise à jour des missions.

1. Méthodes HTTP incompatibles

Problème

- **Backend**: Utilise `@PatchMapping("/{id}")` pour la mise à jour principale des missions
- **Frontend**: Utilise `PUT` dans les méthodes `update` et `updateMission`

Solution recommandée

Deux options possibles:

1. **Modifier le backend** pour accepter les requêtes PUT: Changer `@PatchMapping("/{id}")` en `@PutMapping("/{id}")` dans `MissionControleur.java`
2. **Modifier le frontend** pour utiliser PATCH: Changer les méthodes dans `MissionService.js` pour utiliser `api.patch` au lieu de `api.put`

Exemple de mise à jour du frontend:

```
// Mettre à jour une mission
update: (id, missionData) => api.patch(`${PATH}/${id}`,
missionData).then(response => response.data),
// Mettre à jour une mission (alias pour update)
updateMission: (id, missionData) => api.patch(`${PATH}/${id}`,
missionData).then(response => response.data),
```

2. Problème d'URL pour associer une facture

Problème

- **Backend**: Utilise `/{idMission}/factures/{idFacture}` (pluriel)
- **Frontend**: Utilise `/${missionId}/facture/${factureId}` (singulier)

Solution recommandée

Modifier la méthode `associerFacture` dans le frontend pour utiliser l'URL correcte:

```
associerFacture: (missionId, factureId) =>
api.put(`${PATH}/${missionId}/factures/${factureId}`).then(response =>
```

```
response.data),
```

3. Paramètre supplémentaire "nouvelleAdresse" non géré

Problème

Le backend attend un paramètre optionnel `nouvelleAdresse` qui n'est pas présent dans les appels du frontend.

Solution recommandée

Modifier les méthodes de mise à jour dans le frontend pour inclure ce paramètre:

```
updateMission: (id, missionData, nouvelleAdresse) => {
  let url = `${PATH}/${id}`;
  if (nouvelleAdresse) {
    url += `?nouvelleAdresse=${encodeURIComponent(nouvelleAdresse)}`;
  }
  return api.patch(url, missionData).then(response => response.data);
},
```

Recommandation globale

Pour garantir une compatibilité parfaite entre le frontend et le backend:

1. **Standardiser les méthodes HTTP** en choisissant soit PUT soit PATCH pour toutes les mises à jour (PATCH est recommandé pour des mises à jour partielles)
2. **Standardiser les noms d'URL** en utilisant des conventions cohérentes (pluriel ou singulier)
3. **Documenter l'API complète** avec tous les paramètres possibles pour faciliter l'utilisation correcte par le frontend
4. **Ajouter des tests d'intégration** qui vérifient que les appels frontend fonctionnent correctement avec le backend

Test de la solution

Un script curl pour tester la mise à jour complète d'une mission avec tous les champs:

```
curl -X PATCH "http://localhost:8080/api/missions/1" \
     -H "Content-Type: application/json" \
     -d '{
  "titre": "Mission modifiée avec PATCH",
  "description": "Test de modification via curl - méthode PATCH",
  "dateDebut": "2025-09-01",
  "dateFin": "2025-09-30",
```

```
  "heureDebut": "09:00:00",
  "heureFin": "18:00:00",
  "statutMission": "PLANIFIEE",
  "typeMission": "SURVEILLANCE",
  "nombreAgents": 5,
  "quantite": 15,
  "tarifMissionId": 2,
  "siteId": 2,
  "montantHT": 2500.00,
  "montantTVA": 500.00,
  "montantTTC": 3000.00,
  "commentaires": "Mission modifiée via curl pour test backend"
}' \
     --verbose
```