```java
package com.bnpp.pf.pfspringpoc.feature.notifcase.controllers;

import
com.bnpp.pf.pfspringpoc.feature.notifcase.web.dto.CreateCaseWithNotificationReque
import com.bnpp.pf.pfspringpoc.feature.notifcase.domain.CaseEntity;
import
com.bnpp.pf.pfspringpoc.feature.notifcase.services.NotifCaseFacadeService;
import lombok.RequiredArgsConstructor;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/cases")
@RequiredArgsConstructor
public class NotifCaseFacadeController {

    private final NotifCaseFacadeService service;

    /**
     * Créer un Case + Notification dans une seule transaction.
     */
    @PostMapping("/with-notification")
    public ResponseEntity<CaseEntity> createCaseWithNotification(
            @RequestBody CreateCaseWithNotificationRequest request) {
        CaseEntity created = service.createCaseWithNotification(request);
        return ResponseEntity.ok(created);
    }

    /**
     * Mettre à jour un Case + Notification dans une seule transaction.
     */
    @PutMapping("/{caseId}/with-notification")
    public ResponseEntity<CaseEntity> updateCaseWithNotification(
```

```
            @PathVariable("caseId") UUID caseId,
            @RequestBody CreateCaseWithNotificationRequest request) {
        CaseEntity updated = service.updateCaseWithNotification(caseId,
request);
        return ResponseEntity.ok(updated);
    }
}
```

🧠 `NotifCaseFacadeService.java`

java

```java
package com.bnpp.pf.pfspringpoc.feature.notifcase.services;

import com.bnpp.pf.pfspringpoc.feature.notifcase.domain.*;
import
com.bnpp.pf.pfspringpoc.feature.notifcase.web.dto.CreateCaseWithNotificationReque
import com.bnpp.pf.pfspringpoc.feature.notifcase.web.dto.NotificationDTO;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.UUID;

@Service
@RequiredArgsConstructor
public class NotifCaseFacadeService {

    private final CaseRepository caseRepository;
    private final NotificationRepository notificationRepository;

    @Transactional
```

```
    public CaseEntity
createCaseWithNotification(CreateCaseWithNotificationRequest request) {
        // Mapper le DTO de notification vers l'entité
        NotificationEntity notif =
mapToNotificationEntity(request.getNotification());
        notificationRepository.save(notif);

        // Mapper le DTO de case vers l'entité
        CaseEntity caseEntity = mapToCaseEntity(request);
        caseEntity.setNotification(notif);

        return caseRepository.save(caseEntity);
    }

    @Transactional
    public CaseEntity updateCaseWithNotification(UUID caseId,
CreateCaseWithNotificationRequest request) {
        CaseEntity existingCase = caseRepository.findById(caseId)
                .orElseThrow(() -> new RuntimeException("Case not found with
id: " + caseId));

        // Mise à jour de la notification liée
        NotificationEntity notif = existingCase.getNotification();
        updateNotificationEntity(notif, request.getNotification());
        notificationRepository.save(notif);

        // Mise à jour des champs du Case
        existingCase.setCaseName(request.getCaseName());
        existingCase.setCountry(request.getCountry());
        existingCase.setDataset(request.getDataset());
        existingCase.setLanguage(request.getLanguage());
        existingCase.setActive(request.getActive());
        existingCase.setRetention(request.getRetention());
        existingCase.setAlgorithm(Enum.valueOf(Algorithm.class,
```

```
request.getAlgorithm().toUpperCase()));
        existingCase.setMode(Enum.valueOf(ConfigAuthMode.class,
request.getMode().toUpperCase()));
        existingCase.setTrustedCertificate(request.getTrustedCertificate());

        return caseRepository.save(existingCase);
    }

    private NotificationEntity mapToNotificationEntity(NotificationDTO dto) {
        NotificationEntity n = new NotificationEntity();
        n.setNotificationMode(dto.getNotificationMode());
        n.setCallbackURL(dto.getCallbackURL());
        n.setAuthMode(dto.getAuthMode());
        n.setAuthTokenURL(dto.getAuthTokenURL());
        n.setTopicURL(dto.getTopicURL());
        return n;
    }

    private void updateNotificationEntity(NotificationEntity n,
NotificationDTO dto) {
        n.setNotificationMode(dto.getNotificationMode());
        n.setCallbackURL(dto.getCallbackURL());
        n.setAuthMode(dto.getAuthMode());
        n.setAuthTokenURL(dto.getAuthTokenURL());
        n.setTopicURL(dto.getTopicURL());
    }

    private CaseEntity mapToCaseEntity(CreateCaseWithNotificationRequest req)
{
        CaseEntity c = new CaseEntity();
        c.setCaseName(req.getCaseName());
        c.setCountry(req.getCountry());
        c.setDataset(req.getDataset());
        c.setLanguage(req.getLanguage());
```

```
        c.setActive(req.getActive());
        c.setRetention(req.getRetention());
        c.setAlgorithm(Enum.valueOf(Algorithm.class,
req.getAlgorithm().toUpperCase()));
        c.setMode(Enum.valueOf(ConfigAuthMode.class,
req.getMode().toUpperCase()));
        c.setTrustedCertificate(req.getTrustedCertificate());
        return c;
    }
}
```

🗃️ Repositories (si pas encore créés)

java

```java
package com.bnpp.pf.pfspringpoc.feature.notifcase.domain;

import org.springframework.data.jpa.repository.JpaRepository;
import java.util.UUID;

public interface NotificationRepository extends
JpaRepository<NotificationEntity, UUID> { }
```

java

```java
package com.bnpp.pf.pfspringpoc.feature.notifcase.domain;

import org.springframework.data.jpa.repository.JpaRepository;
import java.util.UUID;

public interface CaseRepository extends JpaRepository<CaseEntity, UUID> { }
```

✅ Résumé des endpoints

Méthode	Endpoint	Description
`POST`	`/api/cases/with-notification`	Crée un `Case` et un `Notification` ensemble
`PUT`	`/api/cases/{caseId}/with-notification`	Met à jour un `Case` et son `Notification` ensemble

Souhaites-tu que je t'ajoute aussi un **mapper avec MapStruct** (pour éviter le mapping manuel entre DTO et Entity) ?
Cela rendrait ton code plus propre et maintenable.