# Tutoriel : Notifications en temps réel entre FastAPI et Flutter

## 1. Introduction

Dans ce tutoriel, nous allons implémenter un système de notifications en temps réel entre une API FastAPI et une application Flutter. Lorsqu'une tâche (todo) est créée sur l'API, une notification sera envoyée à l'application Flutter connectée.

## 2. Utiliser WebSocket pour la communication en temps réel

FastAPI prend en charge les WebSockets nativement. Nous allons créer un point de terminaison WebSocket pour permettre aux clients Flutter de se connecter et recevoir des notifications.

### Exemple de configuration du WebSocket dans FastAPI :

from fastapi import FastAPI, WebSocket, WebSocketDisconnect  
from typing import List  
  
app = FastAPI()  
  
class ConnectionManager:  
 def \_\_init\_\_(self):  
 self.active\_connections: List[WebSocket] = []  
  
 async def connect(self, websocket: WebSocket):  
 await websocket.accept()  
 self.active\_connections.append(websocket)  
  
 def disconnect(self, websocket: WebSocket):  
 self.active\_connections.remove(websocket)  
  
 async def send\_message(self, message: str):  
 for connection in self.active\_connections:  
 await connection.send\_text(message)  
  
manager = ConnectionManager()  
  
@app.websocket("/ws/notifications")  
async def websocket\_endpoint(websocket: WebSocket):  
 await manager.connect(websocket)  
 try:  
 while True:  
 await websocket.receive\_text()  
 except WebSocketDisconnect:  
 manager.disconnect(websocket)

## 3. Envoyer une notification lors de la création d'une tâche

Nous modifions le point de terminaison POST pour envoyer une notification via le WebSocket lorsqu'une todo est créée.

from pydantic import BaseModel  
  
class TodoItem(BaseModel):  
 title: str  
 description: str = ""  
 completed: bool = False  
  
todos = []  
  
@app.post("/todos/")  
async def create\_todo(todo: TodoItem):  
 todos.append(todo)  
 message = f"Nouvelle tâche créée: {todo.title}"  
 await manager.send\_message(message)  
 return {"message": "Todo created", "todo": todo}

## 4. Intégration dans Flutter

Dans l'application Flutter, nous utilisons la bibliothèque `web\_socket\_channel` pour établir une connexion au WebSocket.

### Ajouter `web\_socket\_channel` dans `pubspec.yaml` :

dependencies:  
 web\_socket\_channel: ^2.4.0

### Code Flutter pour la gestion des notifications :

import 'package:flutter/material.dart';  
import 'package:web\_socket\_channel/web\_socket\_channel.dart';  
  
void main() {  
 runApp(MyApp());  
}  
  
class MyApp extends StatelessWidget {  
 @override  
 Widget build(BuildContext context) {  
 return MaterialApp(  
 home: NotificationPage(),  
 );  
 }  
}  
  
class NotificationPage extends StatefulWidget {  
 @override  
 \_NotificationPageState createState() => \_NotificationPageState();  
}  
  
class \_NotificationPageState extends State<NotificationPage> {  
 late WebSocketChannel channel;  
 List<String> notifications = [];  
  
 @override  
 void initState() {  
 super.initState();  
 channel = WebSocketChannel.connect(Uri.parse('ws://127.0.0.1:8000/ws/notifications'));  
 channel.stream.listen((message) {  
 setState(() {  
 notifications.add(message);  
 });  
 });  
 }  
  
 @override  
 void dispose() {  
 channel.sink.close();  
 super.dispose();  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(title: Text("Notifications")),  
 body: ListView.builder(  
 itemCount: notifications.length,  
 itemBuilder: (context, index) {  
 return ListTile(  
 title: Text(notifications[index]),  
 );  
 },  
 ),  
 );  
 }  
}

## 5. Résultat attendu

1. Lorsqu'une tâche est créée via le point de terminaison FastAPI (`/todos/`), un message de notification est diffusé via le WebSocket.  
2. L'application Flutter, connectée au WebSocket, reçoit la notification et l'affiche dynamiquement.

## 6. Remarques

• \*\*Gestion des erreurs :\*\* Implémentez des mécanismes pour gérer les déconnexions ou les erreurs côté client et serveur.  
• \*\*Déploiement sécurisé :\*\* Utilisez HTTPS et WSS en production pour sécuriser les communications.  
• \*\*Scalabilité :\*\* Si vous prévoyez de nombreux clients, envisagez d'utiliser un gestionnaire comme Redis Pub/Sub pour diffuser efficacement les messages.