# FISA — Web Notifications (SignalR via Outbox) — Dev Guide v1.0

**Mục tiêu**: Hướng dẫn Dev BE (.NET 8 + Dapper + PostgreSQL) lập trình **notifications realtime trên website** bằng **SignalR**, **dựa trên kiến trúc Outbox** và **schema v3.1c**. Tài liệu liệt kê **các việc cần làm**, **cách làm**, **ví dụ cụ thể**, để Dev ~1 năm kinh nghiệm có thể triển khai ngay.

## 0) Tổng quan kiến trúc

**Luồng A (đã thống nhất):** Domain Service ghi sự kiện vào public.outbox\_events (trong cùng transaction). **Outbox Worker** đọc outbox theo batch (ORDER BY id, FOR UPDATE SKIP LOCKED), sau đó **push trực tiếp** vào **SignalR Hub** bằng IHubContext<NotifyHub>. FE web kết nối Hub, lắng sự kiện notify để cập nhật Bell Center.

**Nguyên tắc cốt lõi:** - Outbox là **nguồn chân lý** (source of truth) cho event; realtime chỉ **best-effort**. Nếu push SignalR lỗi tạm thời, dữ liệu vẫn hiện qua API Bell Center (pull). - **Group theo user**: server add connection vào group user:{user\_id} để phát đúng người. - **Envelope thống nhất** giữa web và mobile: FE hiển thị theo title/body/deep\_link/severity/...; chi tiết sâu tải bằng API khi người dùng click.

## 1) Chuẩn bị & Phụ thuộc

* .NET 8 WebAPI (Hosting Minimal hoặc Controller-based đều được).
* Gói NuGet: Microsoft.AspNetCore.SignalR, Microsoft.AspNetCore.SignalR.Core (nếu cần), Dapper, Npgsql.
* JWT Authentication đã cấu hình sẵn cho API; Hub sẽ tái sử dụng.
* PostgreSQL schema v3.1c có bảng public.outbox\_events với các cột (gợi ý chuẩn chung):
  + id BIGSERIAL PRIMARY KEY
  + event\_type TEXT NOT NULL
  + payload JSONB NOT NULL (chứa Envelope dạng JSON)
  + created\_at TIMESTAMPTZ NOT NULL DEFAULT now()
  + published\_at TIMESTAMPTZ NULL
  + failed\_attempts INT NULL
  + last\_error TEXT NULL
  + next\_retry\_at TIMESTAMPTZ NULL
  + Các index gợi ý: (published\_at NULLS FIRST, id), partial index cho pending (published\_at IS NULL), index cho next\_retry\_at.

**Lưu ý**: Nếu schema hiện tại thiếu một vài cột retry/backoff, thêm migration SQL ở mục **Phụ lục A**.

## 2) Hợp đồng dữ liệu — Envelope (chuẩn hiển thị)

**Mục đích**: thống nhất dữ liệu FE dùng để render (web + mobile).

**Model C#**

public sealed class Envelope  
{  
 public Guid EventId { get; init; }  
 public string EventType { get; init; } = default!; // e.g. "work\_item.comment\_added"  
 public string Title { get; init; } = default!;  
 public string Body { get; init; } = default!;  
 public string DeepLink { get; init; } = default!; // route nội bộ FE  
 public string Severity { get; init; } = "info"; // info|warning|danger|success  
 public Guid? ProjectId { get; init; }  
 public Guid? WorkItemId { get; init; }  
 public Guid[] Recipients { get; init; } = Array.Empty<Guid>();  
 public object? Extras { get; init; } // SLA, comment snippet, etc.  
 public DateTimeOffset OccurredAt { get; init; }  
}

**Ví dụ payload JSON lưu trong outbox\_events.payload:**

{  
 "EventId": "8a2c2c2c-4f3c-4a3f-9a77-35f65a3f2f11",  
 "EventType": "work\_item.comment\_added",  
 "Title": "Bình luận mới trên Ticket #CR-1024",  
 "Body": "Nguyễn Văn A: Kiểm tra lại log máy chủ…",  
 "DeepLink": "/crm/tickets/83f9-...-1122?tab=comments",  
 "Severity": "info",  
 "ProjectId": "a1b2c3d4-...",  
 "WorkItemId": "83f9e2...1122",  
 "Recipients": ["b2c3d4e5-...-7788", "c3d4e5f6-...-8899"],  
 "Extras": { "commentId": "55f1...9a", "authorId": "aa77...", "createdAt": "2025-10-29T01:33:00Z" },  
 "OccurredAt": "2025-10-29T01:33:00Z"  
}

## 3) Lập trình SignalR Hub

**Bước 3.1 — Đăng ký SignalR trong Program.cs**

var builder = WebApplication.CreateBuilder(args);  
  
// Auth (đang dùng trong API)  
builder.Services.AddAuthentication(/\*...\*/).AddJwtBearer(/\*...\*/);  
  
// SignalR  
builder.Services.AddSignalR();  
  
// Options + Worker  
builder.Services.AddSingleton(new OutboxOptions { BatchSize = 100, PollIntervalMs = 300 });  
builder.Services.AddHostedService<OutboxWorker>();  
  
var app = builder.Build();  
app.UseAuthentication();  
app.UseAuthorization();  
  
// Map Hub  
app.MapHub<NotifyHub>("/hubs/notify");  
  
app.Run();

**Bước 3.2 — Tạo NotifyHub.cs**

using Microsoft.AspNetCore.SignalR;  
  
public class NotifyHub : Hub  
{  
 public override async Task OnConnectedAsync()  
 {  
 var userId = Context.User?.FindFirst("sub")?.Value; // hoặc claim "user\_id"  
 if (!string.IsNullOrEmpty(userId))  
 {  
 await Groups.AddToGroupAsync(Context.ConnectionId, $"user:{userId}");  
 }  
 await base.OnConnectedAsync();  
 }  
}

**Yêu cầu**: Hub chỉ chấp nhận connection có JWT hợp lệ. FE sẽ gắn Bearer token hoặc cookie như API.

## 4) Outbox Worker — Đọc & phát vào SignalR

**Bước 4.1 — Tạo Envelope.cs (mục 2) và OutboxOptions**

public sealed class OutboxOptions  
{  
 public int BatchSize { get; set; } = 100;  
 public int PollIntervalMs { get; set; } = 300;  
}

**Bước 4.2 — Tạo OutboxWorker.cs**

using System.Data;  
using Dapper;  
using Microsoft.AspNetCore.SignalR;  
using Npgsql;  
  
public sealed class OutboxWorker : BackgroundService  
{  
 private readonly string \_connStr;  
 private readonly IHubContext<NotifyHub> \_hub;  
 private readonly OutboxOptions \_opt;  
 private readonly ILogger<OutboxWorker> \_log;  
  
 public OutboxWorker(IConfiguration cfg, IHubContext<NotifyHub> hub,  
 OutboxOptions opt, ILogger<OutboxWorker> log)  
 { \_connStr = cfg.GetConnectionString("MainDb")!; \_hub = hub; \_opt = opt; \_log = log; }  
  
 protected override async Task ExecuteAsync(CancellationToken ct)  
 {  
 while (!ct.IsCancellationRequested)  
 {  
 try  
 {  
 using var conn = new NpgsqlConnection(\_connStr);  
 await conn.OpenAsync(ct);  
  
 using var tx = await conn.BeginTransactionAsync(ct);  
 var rows = await conn.QueryAsync<(long id, string payload)>(  
 @"SELECT id, payload  
 FROM public.outbox\_events  
 WHERE published\_at IS NULL  
 AND (next\_retry\_at IS NULL OR next\_retry\_at <= now())  
 ORDER BY id  
 FOR UPDATE SKIP LOCKED  
 LIMIT @lim",  
 new { lim = \_opt.BatchSize }, tx);  
  
 foreach (var r in rows)  
 {  
 try  
 {  
 var env = System.Text.Json.JsonSerializer.Deserialize<Envelope>(r.payload);  
 if (env?.Recipients != null)  
 {  
 foreach (var uid in env.Recipients)  
 {  
 await \_hub.Clients.Group($"user:{uid}")  
 .SendAsync("notify", env, ct);  
 }  
 }  
 await conn.ExecuteAsync(  
 @"UPDATE public.outbox\_events  
 SET published\_at = now(), last\_error = NULL, failed\_attempts = 0  
 WHERE id = @id",  
 new { r.id }, tx);  
 }  
 catch (Exception ex)  
 {  
 \_log.LogError(ex, "Publish SignalR failed for outbox id={Id}", r.id);  
 await conn.ExecuteAsync(  
 @"UPDATE public.outbox\_events  
 SET failed\_attempts = COALESCE(failed\_attempts,0)+1,  
 last\_error = @err,  
 next\_retry\_at = now() + (  
 CASE WHEN failed\_attempts IS NULL OR failed\_attempts < 5  
 THEN INTERVAL '30 seconds' ELSE INTERVAL '5 minutes' END)  
 WHERE id = @id",  
 new { id = r.id, err = ex.Message }, tx);  
 }  
 }  
  
 await tx.CommitAsync(ct);  
 if (!rows.Any()) await Task.Delay(\_opt.PollIntervalMs, ct);  
 }  
 catch (TaskCanceledException) { }  
 catch (Exception e)  
 {  
 \_log.LogError(e, "Outbox worker loop error");  
 await Task.Delay(1000, ct);  
 }  
 }  
 }  
}

**Điểm kỹ thuật bắt buộc** - Duy trì **đường thẳng** bằng ORDER BY id và **tranh chấp thấp** bằng FOR UPDATE SKIP LOCKED. - Không rollback published\_at khi SignalR lỗi; coi realtime là **best-effort**. - Có **retry/backoff** nhẹ qua failed\_attempts + next\_retry\_at.

## 5) Mapping Event → Envelope (ví dụ khớp v3.1c)

### 5.1 work\_item.comment\_added

* Nguồn dữ liệu: work\_item\_comments, work\_items, work\_item\_watchers.
* Recipients: assignee, requester, toàn bộ watchers.

**Mapper (ví dụ):**

public static class OutboxMapper  
{  
 public static Envelope WorkItemCommentAdded(Guid evtId, WorkItemComment cmt, Guid[] recipients) =>  
 new Envelope {  
 EventId = evtId,  
 EventType = "work\_item.comment\_added",  
 Title = $"Bình luận mới trên Ticket #{cmt.WorkItemCode}",  
 Body = $"{cmt.AuthorName}: {Trim(cmt.Body, 160)}",  
 DeepLink = $"/crm/tickets/{cmt.WorkItemId}?tab=comments",  
 Severity = "info",  
 ProjectId = cmt.ProjectId,  
 WorkItemId = cmt.WorkItemId,  
 Recipients = recipients,  
 Extras = new { commentId = cmt.Id, cmt.AuthorId, cmt.CreatedAt },  
 OccurredAt = DateTimeOffset.UtcNow  
 };  
 static string Trim(string s, int n) => s.Length <= n ? s : s[..n] + "…";  
}

### 5.2 work\_item.status\_changed

* Nguồn: work\_item\_state\_history.
* Recipients: assignee, requester, watchers.
* Body: “Trạng thái: Open → In Progress bởi Trần B”; Severity: success|info tuỳ luật.

### 5.3 ticket.sla\_breach\_soon

* Nguồn: SLA engine (cảnh báo trước khi breach), lưu thông tin SLA trong Extras (deadline, remain\_seconds…).
* Recipients: assignee, supervisor team.
* Severity: warning.

**Gợi ý SQL lấy recipients**: join watchers + assignee + requester theo work\_items.id.

## 6) Bảo mật, Cấu hình, Vận hành

* **Auth/JWT**: Hub dùng cùng scheme với API. FE phải truyền token hợp lệ.
* **RBAC**: Group user:{user\_id} đảm bảo chỉ người đó nhận. Không gửi theo org/project nếu payload không dành cho broadcast.
* **Health**: thêm endpoint /healthz cho service; log số bản tin/giây, số lỗi, latency outbox→hub.
* **Scale**: 1 node không cần Redis backplane; nhiều node thêm Redis backplane cho Broadcast/Group.
* **Observability**: Serilog console + file; mask PII nếu cần.

## 7) QA — Smoke test checklist

1. **Kết nối**: FE connect /hubs/notify → nhận 200/negotiate OK, có ConnectionId.
2. **Comment mới**: tạo bình luận → có dòng mới trong outbox\_events → web nhận notify < 1s.
3. **Đổi trạng thái**: cập nhật status → assignee/requester nhận notify.
4. **Mất mạng tạm thời**: SignalR auto reconnect, tiếp tục nhận sự kiện mới.
5. **Nhiều recipients**: từng user:{uid} nhận đúng; người không liên quan **không** nhận.
6. **Retry**: simulate lỗi publish 1 lần → kiểm tra failed\_attempts, next\_retry\_at → tự hồi phục.
7. **API Bell Center**: nếu tắt Hub, danh sách notification vẫn hiện qua API (pull).

## 8) Definition of Done (DoD)

* NotifyHub hoạt động, JWT-ok, group theo user:{id}.
* OutboxWorker push SignalR (best-effort), giữ nguyên flow outbox hiện có.
* Envelope thống nhất, FE render được ngay, badge tăng realtime.
* Log/healthz đủ; QA smoke pass 7 mục.

## 9) Rủi ro & Cách giảm thiểu

1. **Không có Redis backplane khi scale nhiều node** → có thể miss connection: **Thêm backplane** nếu scale.
2. **Payload quá nặng** (Extras lớn) → tăng latency: giữ Envelope **mảnh**, chi tiết tải qua deep-link API.
3. **Worker treo** → mất realtime: có **liveness probe** + alert; Bell Center (pull) vẫn hoạt động.

## 10) Phụ lục A — Migration SQL gợi ý cho outbox\_events

-- Các cột retry/backoff (nếu thiếu)  
ALTER TABLE public.outbox\_events  
 ADD COLUMN IF NOT EXISTS failed\_attempts int,  
 ADD COLUMN IF NOT EXISTS last\_error text,  
 ADD COLUMN IF NOT EXISTS next\_retry\_at timestamptz;  
  
-- Index hỗ trợ pending  
CREATE INDEX IF NOT EXISTS idx\_outbox\_pending  
 ON public.outbox\_events (published\_at, id)  
 WHERE published\_at IS NULL;  
  
-- Index hỗ trợ retry theo thời gian  
CREATE INDEX IF NOT EXISTS idx\_outbox\_retry  
 ON public.outbox\_events (next\_retry\_at)  
 WHERE published\_at IS NULL;

## 11) Phụ lục B — Truy vấn Dapper mẫu

// Lấy batch outbox pending (đặt trong transaction, FOR UPDATE SKIP LOCKED)  
const string SqlFetch = @"SELECT id, payload  
 FROM public.outbox\_events  
 WHERE published\_at IS NULL  
 AND (next\_retry\_at IS NULL OR next\_retry\_at <= now())  
 ORDER BY id  
 FOR UPDATE SKIP LOCKED  
 LIMIT @lim";  
  
// Đánh dấu published  
const string SqlMarkPublished = @"UPDATE public.outbox\_events  
 SET published\_at = now(), last\_error = NULL, failed\_attempts = 0  
 WHERE id = @id";  
  
// Ghi lỗi + backoff  
const string SqlMarkFailed = @"UPDATE public.outbox\_events  
 SET failed\_attempts = COALESCE(failed\_attempts,0)+1,  
 last\_error = @err,  
 next\_retry\_at = now() + (  
 CASE WHEN failed\_attempts IS NULL OR failed\_attempts < 5  
 THEN INTERVAL '30 seconds' ELSE INTERVAL '5 minutes' END)  
 WHERE id = @id";

## 12) Phụ lục C — Gợi ý kiểm thử thủ công nhanh (Postman + DevTools)

* **Postman**: Gọi API tạo comment → kiểm tra DB có record outbox (payload.event\_type = 'work\_item.comment\_added').
* **DevTools**: tab Network → kiểm tra WebSocket /hubs/notify có frame notify với Envelope đúng.
* **DB**: SELECT \* FROM public.outbox\_events ORDER BY id DESC LIMIT 5; để xem trạng thái publish/retry.

## 13) Kết luận

Với các bước trên, Dev có thể **cắm SignalR realtime** vào kiến trúc Outbox hiện tại **mà không phá vỡ** schema v3.1c. Khuyến nghị giữ Envelope mảnh, đảm bảo retry nhẹ nhàng, và coi SignalR là kênh realtime hỗ trợ — danh sách vẫn do API Bell Center đảm trách.