YaVoy — Resumen de implementación: Incentivos (Wallet/Rewards) + Seguros (Risk Level + Job Insurance)

Documento de trabajo (actualizado a 15/12/2025)

1. Qué se ha conseguido

Este documento resume lo que ya está creado y probado: **Incentivo FIRST_JOB_HELPER**: al completar el primer trabajo del helper se crea reward (pending) y, al liberarse, pasa a earned, se acredita en wallet y se registra en ledger. **Wallets + Ledger + Rewards**: modelo de saldo + movimientos + recompensas. **Orden visual de sectores** con display_order y **riesgo** con risk_level en sectores y microtareas. **Seguro por trabajo**: RPC voy_apply_job_insurance(job_id) asigna póliza/plan activo según el risk_level de la microtarea y genera/actualiza VoyJobInsurance. **Backfill**: jobs antiguos que tenían category tipo “DIGITAL/MAYORES/HOGAR/errands” se mapearon a microtareas genéricas.

2. Tablas y campos relevantes

Core VoyJobs: añadido microtask_id (text) con FK a VoyMicroTasks(id). VoyJobAssignments: ciclo real del trabajo (ASSIGNED → COMPLETED). Aquí se engancha el incentivo. **Catálogo** VoySectors: display_order (orden UI) y risk_level. VoyMicroTasks: risk_level (ajuste fino por microtarea). **Incentivos** VoyWallets, VoyWalletLedger, VoyRewards. **Seguros** VoyInsuranceProviders, VoyInsurancePolicies, VoyInsurancePlans. VoyJobInsurance: snapshot por job (plan, fee, estado, etc.).

3. SQL aplicado (bloques principales)

3.1. Orden de sectores con display_order

```
ALTER TABLE public."VoySectors"
ADD COLUMN IF NOT EXISTS display_order integer;

UPDATE public."VoySectors" SET display_order = 1  WHERE id = 'mayores-dependencia';
UPDATE public."VoySectors" SET display_order = 2  WHERE id = 'hogar-mantenimiento';
UPDATE public."VoySectors" SET display_order = 3  WHERE id = 'mascotas';
UPDATE public."VoySectors" SET display_order = 99 WHERE id = 'otros-servicios';

SELECT id, name, emoji, description
FROM public."VoySectors"
ORDER BY display_order, name;
```

3.2. Risk level en sectores

```
ALTER TABLE public."VoySectors"
ADD COLUMN IF NOT EXISTS risk_level smallint;

UPDATE public."VoySectors" SET risk_level = 2 WHERE risk_level IS NULL;

ALTER TABLE public."VoySectors"
ADD CONSTRAINT voysectors_risk_level_check
CHECK (risk_level IN (1,2,3));
```

3.3. Risk level en microtareas + herencia desde sector

```
ALTER TABLE public."VoyMicroTasks"
ADD COLUMN risk_level smallint;

UPDATE public."VoyMicroTasks" mt
SET risk_level = s.risk_level
FROM public."VoySectors" s
WHERE mt.sector_id = s.id
  AND mt.risk_level IS NULL;

ALTER TABLE public."VoyMicroTasks"
ADD CONSTRAINT voymicrotasks_risk_level_check
CHECK (risk_level IN (1,2,3));
```

3.4. Mapeo jobs antiguos a microtask_id (genéricos)

```
INSERT INTO public."VoyMicroTasks"(id, sector_id, name)
VALUES ('digital-generico','tecnologia-digital','Digital (genérico)')
ON CONFLICT (id) DO NOTHING;

INSERT INTO public."VoyMicroTasks"(id, sector_id, name)
VALUES ('mayores-generico','mayores-dependencia','Mayores (genérico)')
ON CONFLICT (id) DO NOTHING;

INSERT INTO public."VoyMicroTasks"(id, sector_id, name)
VALUES ('hogar-generico','hogar-mantenimiento','Hogar (genérico)')
ON CONFLICT (id) DO NOTHING;

INSERT INTO public."VoyMicroTasks"(id, sector_id, name)
VALUES ('recados-generico','compras-recados','Recados (genérico)')
ON CONFLICT (id) DO NOTHING;

INSERT INTO public."VoyMicroTasks"(id, sector_id, name)
VALUES ('OTROS','otros-servicios','Otros / A definir')
ON CONFLICT (id) DO NOTHING;

UPDATE public."VoyJobs"
SET microtask_id = CASE
```

```
  WHEN upper(category) = 'DIGITAL' THEN 'digital-generico'
  WHEN upper(category) = 'MAYORES' THEN 'mayores-generico'
  WHEN upper(category) = 'HOGAR'   THEN 'hogar-generico'
  WHEN lower(category) = 'errands' THEN 'recados-generico'
  ELSE 'OTROS'
END
WHERE microtask_id IS NULL;
```

4. Seguro por job: RPC + trigger + query UI

```
-- RPC: aplica seguro a un job en base al risk_level de la microtarea
CREATE OR REPLACE FUNCTION public.voy_apply_job_insurance(p_job_id uuid)
RETURNS void
LANGUAGE plpgsql
SECURITY DEFINER
AS $$
DECLARE
  v_microtask_id text;
  v_risk smallint;
  v_policy_id uuid;
  v_plan_id uuid;
  v_fee numeric(12,2);
BEGIN
  SELECT microtask_id INTO v_microtask_id
  FROM public."VoyJobs" WHERE id = p_job_id;
  IF v_microtask_id IS NULL THEN
    RAISE EXCEPTION 'Job % has no microtask_id', p_job_id;
  END IF;

  SELECT risk_level INTO v_risk
  FROM public."VoyMicroTasks" WHERE id = v_microtask_id;
  IF v_risk IS NULL THEN v_risk := 2; END IF;

  SELECT id INTO v_policy_id
  FROM public."VoyInsurancePolicies"
  WHERE status='ACTIVE'
    AND start_date <= current_date
    AND end_date >= current_date
  ORDER BY start_date DESC
  LIMIT 1;
  IF v_policy_id IS NULL THEN
    RAISE EXCEPTION 'No ACTIVE insurance policy found';
  END IF;

  SELECT id, price_per_job INTO v_plan_id, v_fee
  FROM public."VoyInsurancePlans"
  WHERE policy_id = v_policy_id
    AND is_active=true
    AND risk_level=v_risk
  ORDER BY price_per_job ASC
  LIMIT 1;

  INSERT INTO public."VoyJobInsurance"(
    job_id, policy_id, plan_id, risk_level, insurance_fee, currency, status, created_at, updated_at
  )
  VALUES (p_job_id, v_policy_id, v_plan_id, v_risk, COALESCE(v_fee,0), 'EUR', 'APPLIED', now(), now())
  ON CONFLICT (job_id) DO UPDATE
  SET policy_id=EXCLUDED.policy_id,
      plan_id=EXCLUDED.plan_id,
      risk_level=EXCLUDED.risk_level,
      insurance_fee=EXCLUDED.insurance_fee,
      status=EXCLUDED.status,
      updated_at=now();
END;
$$;

-- Trigger: aplica seguro en insert/update de microtask_id
CREATE OR REPLACE FUNCTION public.voy_trg_apply_job_insurance()
RETURNS trigger
LANGUAGE plpgsql
SECURITY DEFINER
AS $$
BEGIN
  IF NEW.microtask_id IS NOT NULL THEN
    PERFORM public.voy_apply_job_insurance(NEW.id);
  END IF;
  RETURN NEW;
END;
$$;

DROP TRIGGER IF EXISTS trg_voy_jobs_apply_insurance ON public."VoyJobs";
CREATE TRIGGER trg_voy_jobs_apply_insurance
AFTER INSERT OR UPDATE OF microtask_id
ON public."VoyJobs"
FOR EACH ROW
EXECUTE FUNCTION public.voy_trg_apply_job_insurance();

-- Query UI tipo “ticket”
SELECT
  j.id AS job_id,
  j.title,
  j.description,
  j.category,
  j.microtask_id,
  mt.name AS microtask_name,
  mt.risk_level AS microtask_risk_level,
  ji.insurance_fee,
  ji.currency,
  ji.status AS insurance_status
FROM "VoyJobs" j
JOIN "VoyMicroTasks" mt ON mt.id = j.microtask_id
LEFT JOIN "VoyJobInsurance" ji ON ji.job_id = j.id
WHERE j.id = '<JOB_ID>'::uuid;
```

5. Qué falta por hacer (web/admin → luego móvil)

Web / Crear anuncio: UI “ticket” con líneas (precio trabajo, seguro, plataforma, total comisión) + pago Stripe. **Web / Candidatos**: ya creas assignment al aceptar. Falta reforzar UI/estado (ASSIGNED/COMPLETED). **Web / Pago**: al marcar “completado para pagar” actualizar assignment + disparar reward + (si aplica) marcar insurance como PAID/ACTIVE. **Admin**: paneles de Jobs, Seguros (providers/policies/plans), Wallets/Rewards. **Móvil**: replicar pantallas clave cuando web esté estable.