Perfecte 👌, et deixo el **README complet i actualitzat** amb el nou fitxer calculatePersonalNeeded.ts inclòs.

**📘 Autogeneració de Quadrants – Documentació Tècnica**

Aquest document descriu l’arquitectura i el flux de treball del sistema d’**autogeneració de quadrants** a la webapp de **Cal Blay**.  
Es detalla quins fitxers participen en el procés, quina responsabilitat té cadascun i com es comuniquen entre ells.

**🔄 Flux General**

1. **Usuari obre el QuadrantModal** → introdueix dates, hores, nombre de treballadors i conductors, etc.
2. El **frontend (QuadrantModal.tsx)** crida a APIs per obtenir personal i vehicles disponibles.
3. Quan l’usuari prem **“Auto generar i desa”**, s’envien les dades a l’API **/api/quadrants**.
4. L’API orquestra la lògica amb el servei **autoAssign.ts**.
5. autoAssign aplica les regles de:
   * disponibilitat (personnelRest.ts, personnel/available)
   * descans mínim
   * equitat i hores setmanals/mensuals (workloadLedger.ts)
   * premisses especials (premises-<dept>.json)
   * assignació de vehicles i conductors fixos
   * càlcul de treballadors necessaris (calculatePersonalNeeded.ts)
6. El resultat (responsable, conductors i treballadors assignats) es desa a Firestore a la col·lecció **quadrants<Departament>**.

**🗂 Fitxers Principals**

**1. Frontend – Recollida i crida**

**src/app/menu/quadrants/[id]/components/QuadrantModal.tsx**

* Component React que mostra el formulari/modal per crear un quadrant.
* Funcionalitats clau:
  + Selecció de dates i hores.
  + Selecció de nombre de treballadors i conductors.
  + Carrega vehicles disponibles amb /api/transports/available.
  + Carrega personal disponible amb el hook useAvailablePersonnel.
  + Mostra opcionalment selectors per **tipus de vehicle** i **matrícules disponibles**.
  + Envia la sol·licitud a /api/quadrants amb totes les dades.

**src/app/menu/quadrants/[id]/hooks/useAvailablePersonnel.ts**

* Hook React que consulta /api/personnel/available.
* Retorna tres llistes filtrades:
  + responsables
  + conductors
  + treballadors
* S’encarrega de:
  + respectar exclusions (excludeIds)
  + marcar estat available / conflict
  + mostrar logs de depuració

**2. APIs de disponibilitat**

**src/app/api/personnel/available/route.ts**

* Endpoint GET que calcula quins **persones del departament estan lliures** en un rang de temps.
* Regles aplicades:
  + No estar ja assignat en un altre quadrant en el mateix rang.
  + Complir hores mínimes de descans (personnelRest.ts).
  + Pertànyer al departament correcte.
* Retorna tres arrays: responsables, conductors i treballadors.
* Només es retornen en estat available.

**src/app/api/transports/available/route.ts**

* Endpoint POST que comprova quins **vehicles estan lliures** en el rang de temps.
* Revisa la col·lecció transports i totes les col·leccions quadrants\* per detectar solapaments.
* Retorna vehicles amb camps: id, plate, type, available.

**3. Lògica de negoci – Assignació**

**src/services/autoAssign.ts**

* **Peça central** de l’autogeneració.
* Responsabilitats:
  + Carregar premisses del departament (premises-<dept>.json).
  + Construir el **ledger** d’hores (workloadLedger.ts).
  + Filtrar personal disponible i elegible (eligibility.ts).
  + Escollir responsable (manual, per premissa o per ranking).
  + **Assignar conductors:**
    - Si vehicle té conductorId fix → comprovar disponibilitat i hores → assignar si possible.
    - Si no → buscar conductor alternatiu del pool, amb criteris d’equitat.
  + Calcular nombre de treballadors necessaris (calculatePersonalNeeded.ts).
  + Assignar treballadors equilibradament segons càrregues del ledger.
* Retorna:
* {
* assignment: {
* responsible: { name },
* drivers: [{ name, vehiclePlate?, meetingPoint }],
* staff: [{ name, meetingPoint }]
* },
* meta: { needsReview, violations, notes }
* }

**src/utils/calculatePersonalNeeded.ts**

* Helper matemàtic que calcula **quants treballadors reals calen**.
* Fórmula:
* needed = staffCount - (conductorsReals + responsable)
  + Si responsable també és conductor → no es compta doble.
  + Si no hi ha responsable → no es resta.
  + Si el resultat és negatiu → retorna 0.
* Exemple:
* calculatePersonalNeeded({
* staffCount: 5,
* drivers: [{ name: 'Joan' }, { name: 'Pere' }],
* responsableName: 'Joan'
* })
* // Retorna 3 treballadors

**src/utils/personnelRest.ts**

* Helpers per comprovar **descans mínim i solapaments**.
* Funcions:
  + hasMinRestByName → valida descans mínim entre torns.
  + getBusyPersonnelIds → retorna IDs/noms ocupats en un rang.
  + loadMinRestHours → llegeix premisses per departament (hores mínimes de descans).

**src/services/workloadLedger.ts**

* Calcula **equitat i càrregues de treball**:
  + hores setmanals i mensuals per persona
  + nombre d’assignacions
  + últim cop assignat
* S’utilitza a autoAssign.ts per prioritzar qui té menys càrrega acumulada.

**4. Persistència**

**src/app/api/quadrants/route.ts**

* Endpoint POST que rep la sol·licitud d’autogeneració.
* Orquestra la crida a autoAssign(payload).
* Desa el resultat a Firestore a la col·lecció específica (quadrantsLogistica, quadrantsServeis, etc.).
* Retorna la proposta amb responsables, drivers i staff.

**⚙️ Exemple de Flux**

1. Usuari obre modal i demana:
   * 5 treballadors, 2 conductors, dates + hores.
   * Selecciona 2 vehicles.
2. QuadrantModal.tsx fa:
   * GET /api/personnel/available?department=Logistica
   * POST /api/transports/available
3. Usuari clica “Auto generar i desa”:
   * POST /api/quadrants amb department, dates, totalWorkers, numDrivers, vehicles.
4. autoAssign.ts:
   * Troba responsable més adequat.
   * Revisa vehicles i assigna conductors (fixos o alternatius).
   * Calcula i assigna staff equilibradament amb calculatePersonalNeeded.
5. route.ts:
   * Desa a Firestore a quadrantsLogistica/{eventId}.
   * Respon amb proposta generada.

**📌 Notes Importants**

* **Vehicles i conductors fixos**:  
  autoAssign comprova si un vehicle té conductorId. Si aquest conductor està disponible i dins dels límits d’hores, s’assigna prioritàriament.
* **Equitat i hores**:  
  Tot el sistema de repartiment es recolza en workloadLedger.ts, que assegura que no es repeteixin sempre les mateixes persones.
* **Descans mínim**:  
  Sempre comprovat amb personnelRest.ts.
* **Premisses específiques**:  
  Definides per JSON (src/data/premises-<dept>.json), permeten forçar responsables o regles especials per ubicacions.

Vols que et prepari ara un **diagrama visual (flowchart)** del procés per incloure’l al README i que sigui encara més clar per explicar-ho a la direcció i a l’equip tècnic?