Refactor the Request/DraftQuote wizard into a simpler, 3-step, schema-driven flow with autosave and live preview. Keep API contracts unchanged. Remove obsolete files/components. Produce clean commits.

🎯 Goals (must have)

- Reduce wizard to **3 steps**: `Basics` → `Options` (Seafreight/Haulage/Services tabs) → `Review & Send`.
- Central **Zod schema** (`DraftQuoteFormSchema`) for validation & types.
- **react-hook-form** + `zodResolver`, single **WizardEngine** to orchestrate.
- **Autosave** (debounce 800 ms) + **resumeToken** (client-side UUID).
- **LivePreview** side panel (updates in real-time).
- **Adapter** `toDraftQuotePayload()` that maps form → current backend payload (no API contract changes).
- Flat **error panel** + “jump to field” anchors.
- Delete legacy wizard steps/components/services that become redundant.

🔒 Constraints / Invariants

- **Do NOT modify** backend endpoints, models, or scopes.
- Keep **current DraftQuote payload** structure and field names.
- Keep MSAL setup and API clients generated from OpenAPI.
- Use existing design system (MUI/TanStack); no new UI libs.

📁 Target file structure (create/update)

```
features/
  request/
```

```
    wizard/
      schema.ts                 # NEW (Zod)
      wizard.config.ts          # NEW (steps metadata)
      WizardEngine.tsx          # NEW (form provider, autosave,
routing)
      LivePreview.tsx           # NEW (side summary)
      StepRouter.tsx            # NEW (routes → pages)
      toDraftQuote.ts           # NEW (adapter to API payload)
      pages/
        BasicsStep.tsx          # NEW
        OptionsStep.tsx         # NEW (Tabs:
Seafreight/Haulage/Services)
        ReviewStep.tsx          # NEW
        ExpressStep.tsx         # NEW (optional, linked from wizard
entry)
```

If project uses different base path (e.g. `features/quote`), adapt accordingly, but keep the same internal structure.

🧠 1) Schema & Types (Zod)

Create `features/request/wizard/schema.ts`:

- Export `DraftQuoteFormSchema` and `DraftQuoteForm` (inferred type).
- Shape:

```
import { z } from "zod";

export const DraftQuoteFormSchema = z.object({
  basics: z.object({
    cargoType: z.enum(["FCL","LCL","AIR"]),
    incoterm: z.string().min(3),
    origin: z.object({ city: z.string().min(2), country:
z.string().length(2) }),
    destination: z.object({ city: z.string().min(2), country:
z.string().length(2) }),
    requestedDeparture: z.string().optional(), // ISO string
    goodsDescription: z.string().min(2),
  }),
  options: z.object({
    seafreights: z.array(z.object({
      carrier: z.string(),
      service: z.string(),
      etd: z.string().optional(),
      eta: z.string().optional(),
      rates: z.array(z.object({
        containerType: z.string(),
        basePrice: z.number().nonnegative(),
      })).min(1)
    })).default([]),
    haulages: z.array(z.object({
      mode: z.enum(["truck","rail","barge"]),
      leg: z.enum(["pre","on","post"]),
      price: z.number().nonnegative(),
      note: z.string().optional()
    })).default([]),
    services: z.array(z.object({
      code: z.string(),
      label: z.string(),
      price: z.number().nonnegative().optional()
    })).default([]),
  }),
  attachments: z.array(z.object({
    name: z.string(),
    url: z.string().url()
  })).default([]),
});
export type DraftQuoteForm = z.infer<typeof DraftQuoteFormSchema>;
```

⚙️ 2) Steps config

Create `features/request/wizard/wizard.config.ts`:

```
export const wizardSteps = [
  { id: "basics",  label: "Basics",         fields: ["basics.*"] },
  { id: "options", label: "Options",        fields: ["options.*"] },
  { id: "review",  label: "Review & Send", fields:
["basics.*","options.*","attachments"] },
] as const;

export type WizardStepId = typeof wizardSteps[number]["id"];
```

🧩 3) Wizard Engine

Create `features/request/wizard/WizardEngine.tsx`:

- `react-hook-form` with `zodResolver`.
- Autosave debounced (800 ms) via `onAutoSave(values)`.
- Layout: main content + `LivePreview` sidebar + footer nav.
- Provide `FormProvider`.
- Integrate `StepRouter`.

Implement a default export component:

```
// props: defaultValues, onAutoSave(form), onSubmit(payload)
```

🧭 4) Step Router

Create `features/request/wizard/StepRouter.tsx`:

- Accepts `currentStep` (from route or internal state) and renders:

    - `BasicsStep` / `OptionsStep` / `ReviewStep`.
- Use `useFormContext()` to pass form methods to steps.

👁️ 5) Live Preview

Create `features/request/wizard/LivePreview.tsx`:

- Props: `values: DraftQuoteForm`.
- Show concise summary: origin → destination, incoterm, cargoType, counts of seafreights/haulages/services, and (if available) computed estimated total (sum of basePrice + options with naive calculation; do not persist this value).

🔀 6) Adapter to API

Create `features/request/wizard/toDraftQuote.ts`:

- Export `toDraftQuotePayload(f: DraftQuoteForm)` → returns **existing** DraftQuote payload shape used today (no changes to backend contract).
- Include `resumeToken` (uuid v4) client-side.
- Keep `status: "in_progress"` and `version: 1`.

🧱 7) Pages

Create pages:

`BasicsStep.tsx`

- Form fields for basics (MUI + RHF controllers).

- Validate via schema (errors under fields).

`OptionsStep.tsx`

- MUI Tabs: **Seafreight / Haulage / Services**.
- Each tab uses a TanStack Table or simple list with **Add**, **Edit**, **Remove**.
- Keep state in RHF arrays (`useFieldArray`).

`ReviewStep.tsx`

- Read-only summary of all sections + attachments manager.
- Submit → calls `onSubmit(toDraftQuotePayload(values))`.

`ExpressStep.tsx` (optional)

- 5 questions (cargoType, origin, destination, incoterm, goodsDescription).
- On submit, pre-fill form and navigate to `review`.

💾 8) Autosave & Guards

- In `WizardEngine`, implement `form.watch` + debounce 800 ms → `onAutoSave(values)`.
- Add `beforeunload` guard if form is dirty and last autosave > 2s.
- Build a flat error panel component listing all errors with anchors; clicking scrolls to the field.

🧹 9) Remove legacy/unused files

Delete old wizard steps/components/services no longer used. Search and remove (case-insensitive) in `features/request/**` and `features/quote/**`:

- Files matching (wildcards):
    - `*CargoStep*.tsx`, `*DatesStep*.tsx`, `*IncotermStep*.tsx`
    - `*SeafreightStep*.tsx`, `*HaulageStep*.tsx`, `*ServicesStep*.tsx`
    - `*AttachmentsStep*.tsx`, `*SummaryStep*.tsx`, `*ConfirmationStep*.tsx`
    - Any previous `Wizard.tsx` that duplicates flow logic
    - Legacy adapters that map UI → API but are superseded by `toDraftQuote.ts`
- Remove dead hooks under `features/**/hooks/` that belong to deleted steps.
- Remove dead route entries from router config that point to deleted pages.
- Remove CSS/modules only referenced by deleted files.

Caution:

- Do not delete shared components used elsewhere (`AutocompleteCity`, `IncotermSelect`, etc.) unless they're **exclusively** referenced by deleted steps. Use Cursor's "find references" to confirm.

🧰 10) Wiring / Routing

- Update `react-router-dom` routes to point the wizard entry to the new `WizardEngine` + `StepRouter`.
- Keep route params/paths identical from the outside (backward compatible URLs), but internally map to `basics`, `options`, `review`.
- Add optional route for `/request/express` → `ExpressStep`.

🧪 11) Non-regression test checklist (create minimal tests if present setup allows)

If `vitest` is configured, add tests under `features/request/wizard/__tests__/`:

- `toDraftQuote.spec.ts`: ensures payload fields (incoterm, cargoType, origin/destination city/country, options arrays) match current backend expectations.
- `schema.spec.ts`: validates a good sample and fails on missing required fields.

If tests infra not present, skip but keep the checklist in a `README.md`.

🧾 12) Linting & Rules

- Ensure ESLint runs; add a local rule (or comment) to discourage importing SDK types directly in UI files. UI should consume only internal types from `schema.ts`.
- Fix imports after deletions.

🚀 13) Acceptance criteria

- Build succeeds: `vite build`.
- Type-check passes.
- Navigating wizard requires **≤ 3 steps** (plus optional Express).
- Filling **Basics** and adding **at least 1 seafreight rate** makes **Review** valid.
- Submit calls existing create/update endpoints with **unchanged** payload format.
- Autosave triggers network updates (or local persistence) without changing backend contract.

- No dead routes/imports; no TS or ESLint “unused” warnings from deleted files.

🧭 14) Sample commit plan (please follow)

1. `feat(wizard): add schema, config, engine, router, pages, preview, adapter`
2. `refactor(wizard): wire routes and autosave, add error panel and guards`
3. `chore(wizard): remove legacy steps/components and dead routes`
4. `test(wizard): add payload and schema tests (if test runner available)`
5. `docs(wizard): add README with usage and non-regression checklist`

📌 Notes

- Do not touch MSAL setup or OpenAPI clients.
- Keep APIM/base URLs resolution as is.
- If any old step contains unique business rules, **move** that logic into:
  - the Zod schema (validation), or
  - `toDraftQuote.ts` (mapping), or
  - a small helper in `features/request/wizard/utils.ts`.

End of instructions. Apply all changes now.