Here are the main points from the list that I haven't fully covered yet, and how I'd handle them.

Architecture / organization

Right now the code is split into components, composables, and an API layer, which is fine for a small app. If this grew, I'd be stricter about separation:

- pure calculation logic stays completely isolated (easy to test, no side effects)
- API functions stay thin and only deal with request/response shaping + errors
- async state (loading/error/data) lives in a composable or store so the UI stays clean

Store usage

Location-wise, it's hardcoded to Berlin for now, but a next step would be making it dynamic:

- optionally detect the region via browser geolocation as a default suggestion
- add a region select/input so the user can change it and instantly see updated taxes and results
- store the selected region in Pinia (and optionally persist it) so it survives refreshes

Browser/device support

I haven't explicitly documented this, but I'd target modern browsers (Chrome/Firefox/Edge/Safari) and mobile Safari/Chrome.

Improving test effectiveness + coverage

My current tests cover the happy paths, but to make them more effective I'd add:

- reactivity tests (changing refs updates computed values)
- edge cases (commission on/off, zero price, extreme values, rounding behavior)
- API failure tests (network errors, missing fields in the response)
- a couple integration tests for the form flow (valid input)

The goal is tests that catch real mistakes, not just "it returns something".

CORS and production endpoints

In development I'd handle CORS using a Vite proxy so the browser never hits the external endpoint directly. In production, the correct solution is usually a backend proxy/serverless function (so you avoid CORS issues completely, can hide secrets, and optionally cache responses).

Slow API responses

Right now I show loading/error states, but I'd go further if the endpoint is slow:

- cache tax results (they don't change often)
- prevent duplicate requests while one is in flight
- add retry + timeout handling