ESSENTIAL TASKS

Game map

- ✔️Game map is a grid with a random number of rows and columns (2 < N <= 20).
- ✔️Each tile is one of 3 types, chosen randomly: Forest, Wasteland or City.
- ✔️Each tile has a background image; take it from `tiles.json`.
- ⚠️When the app is launched for the 1st time, the map is generated automatically.
  Comment: Meteor publication absolutely shouldn't contain this logic. It should be moved to a separate file (sometimes called *fixtures.js*). It should happen on the server only, Meteor method call is not needed here.
- ✔️ User can remove the old map and generate a new one by clicking the "Reset" button.

Pollution

- ✔️There is no pollution in the beginning.
- ❌Each City emits `P_emit` units of pollution every `T_emit` seconds.
  Comment: the timer doesn't work properly. I set a 10 seconds interval and I still see generated pollution every second. The client shouldn't use any timers, everything should happen on the server.
- ❌Pollution is distributed randomly between adjacent tiles according to the following rules:
  i. ✔️Wasteland and City can hold an unlimited amount of pollution.
  ii. ✔️Forest can hold `P_forest` units of pollution. If this limit is exceeded, Forest becomes a Wasteland.
  Comment 1: pollution generated by the City shouldn't pollute this City, it should go immediately to the adjacent tile. A City can be only polluted by another City if they are located on adjacent tiles.
  Example: City-1 generates 4 pollution units. 2 units go to the adjacent Wasteland, 1 unit goes to the adjacent Forest, 1 unit goes to the adjacent City-2, 0 units go to the City-1.
  Comment 2: only the tiles near the Cities can be polluted. Those tiles that don't border any city can never be polluted.
  Comment 3: please, double-check the pollution logic after implementing the changes. It looks like there are some bugs.

User input

- ⚠️User can click the "Reset" button to generate a new map.
  ⚠️User can manipulate the parameters mentioned in the Pollution section. You (developer) should set some sensible defaults and limits.
- ⚠️New values are applied immediately. Existing timers are not affected.

Comment 1: pollution generation should start immediately after the reset.
Comment 2: input values shouldn't be reset with the map.
Comment 3: if I switch browser tabs, the pollution logic stops working properly: the existing inputs are ignored, the new inputs aren't applied. I should reset the map to make it work.

UI

- The header contains:
  - ~~amount of groundwater pollution,~~
    Comment: please, remove it. It was my mistake.
  - ✔️"Reset" button,
  - ✔️parameters inputs.
- ❌ The map is located below the header. It always fits the width of the container. It may have a vertical scrollbar if necessary.
  Comment: the map should **always** fit the width of the container. There should be no empty margins on the left and right, even if the map is narrow.
- ✔️Each tile has a background image and a number of pollution.
- ❌Tile's aspect ratio is defined by its background image aspect ratio (the same for all tiles).
  Comment: the tiles shouldn't have fixed 1/1 aspect ratio, their aspect ratio should be defined by their background image. The background image should not be cropped or stretched. Also, the tiles should not have fixed width.
- ✔️The layout must be responsive.

BONUS TASKS

- ❌The map always covers all available space below the header. It fits either width or height, depending on its aspect ratio. The other side may have a scrollbar if necessary.
- ✔️Add some nice styling for the header.

Other comments and questions:

1. You put all tiles into one Mongo document and use *nanoid* to give them ids. Why? I see no reason for that. The natural solution is to use a separate MongoDB collection for tiles. Ids are generated automatically for every document, there is no need for the external library.
2. The only Meteor methods you need to define are "*map.generate"* and *"map.onValueChange"*. All other stuff should happen on the server only, you shouldn't initiate pollution and distribution logic from the client.
3. Why have you used *laravel-mix* and *cssnano*? Meteor handles build and minification steps by itself. Maybe because of TailwindCSS JIT mode?
4. You've used Typescript, although we didn't ask for that.