Pension Pots API

Plan

1. setup postgres with docker for local development
2. Setup project using typeorm starter script
3. spilt data into three tables `pension_pots`, `pension_providers`, `searched_pension` to reduce redundency
4. populate db
5. build out endpoints
6. `/pension-pots`
7. `/searched-pensions`
8. `/pots`
9. `/pots/search?potName=Google`
10. `/pots/search?id=:id`
11. `/pots/search/:value?greaterThan=:number`
12. `/pots/search/:value?lessThan=:number`
13. `/searched-pensions/found`
14. `/pots/search?employer=:employer`
15. `/pots/search?pensionProvider=:pensionProvider.name`
16. `/pots?forecast=true`
17. write acceptatance tests for each endpoint

Technology

- nodejs
- TypeScript
- Express.js
- Vitest
- PostgreSQL
- Docker

DB structure

Tradeoffs

- **Typeorm starter** - Choose to use typeorm becuase have recently inherited a project at work that uses it and wanted to build something from scratch to improve my understanding of it. I decided to use the starter script `typeorm init --database postgres --express` owing to time limits. It got me up and running quickly but dependencies where a little outdated along with the TypeScript config. With more time, I'd prefer to modify the config or set up the project from scratch. Also Typeorm itself didn't play nicely with the test setup I am used to so stopped me from getting going with that.

Whats next?

- fix and write more tests
- refactor code so organised according to components - `pensionPots` & `searchedPensions`
- extract repetative code in reusable utils - e.g. `flattenSearchedPensions`
- For the search endpoint on each search update the `searched_pensions` table
- remove extraineous data from response e.g. `searchedPensions` or `id` on pension providers
-