Corti QA Engineering Test: Solution

What was implemented:

- The solution showcases testing of the ***pet*** API endpoints for:
  - POST /pet
  - POST /pet{petId}/uploadImage
  - PUT /pet
  - DELETE /pet/{petId}

ToDo: add coverage for the other endpoints & API sections inside the swagger documentation.

Tool used:

- The Playwright test automation framework: https://playwright.dev/.

What was implemented:

- Testing valid, non-valid, and unsupported ID values and file types.

- Tests parameterisation:
  - Running the same test multiple times with different inputs.

- Testing flows:
  - Implemented one happy path flow: create pet, upload image to pet, update pet info, delete pet.

- Implementation-wise, I used the ***describe*** interface from Playwright using ***test*** and ***test.step*** annotations. In certain scenarios we might need to use the ***beforeEach***, ***beforeAll***, ***afterEach***, ***afterAll*** test hooks.

Bonus:

- CI/CD
  - The current implementation is able to be set & run via a CI/CD tool, eg. TeamCity. I foresee the following steps in the setup:
    1. The CI/CD server has access to clone the test repository
    2. Framework setup/installation via ***npm install***
    3. Running test: ***npx playwright test***
    4. We can have a bunch of other optional steps depending of the setup:
       - We might want to setup/reset a DB before running tests
       - We might need to spin up an environment
       - A custom reporting step

- Other configurations:
  - Automated triggers after a successful deployment of the app-in-test to a test environment.
  - Parameters for the test job: test environment name, etc.

- Test rerun-ability & repeatability:
  - As the good practices of test automation recommend, the tests should be re-runnable, independent from each other, focused to a single feature of the application under test (*all these, as much as possible).
  - In order to have re-runnable tests it is important to have a mechanism that handles the test data.
  - Example of such mechanisms:
    - Database reset to a known starting state (with pre-populated data in it)
    - On-the-fly test data creation before each test or test suite.
    - The test data handling mechanism is vital for the tests' assertions to pass.

- Reporting:
  - Playwright default HTML reporter
  - Slack or MS Teams per team channel notifications
  - In CI/CD, eg. TeamCity, the report can be saved in the Artifacts (test runs can also be followed via the Build Log)
  - 3rd-party reporters for Playwright, eg. Monocart, Tesults, ReportPortal, etc.