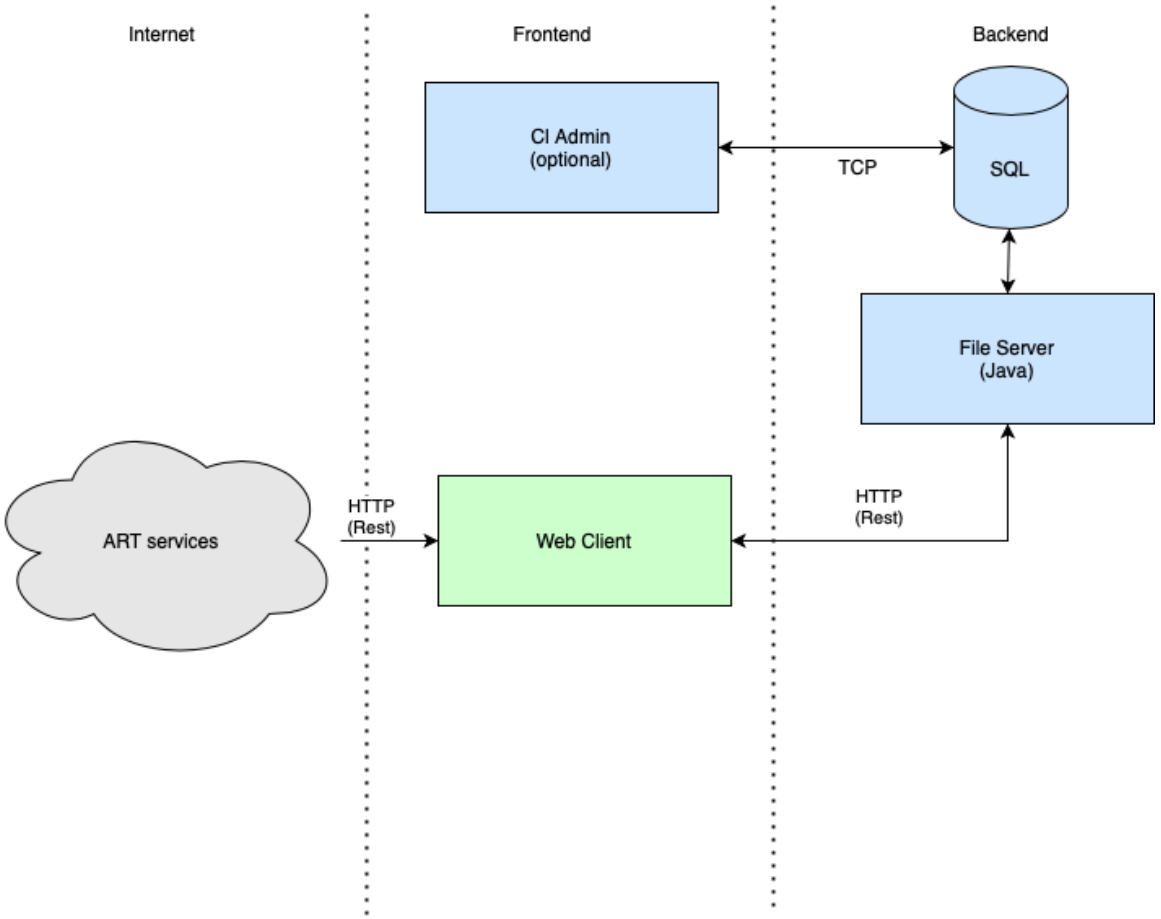
# Artworks

## Story

Your millionaire art connoisseur friend has hired you and your team, to create a website for them. Your team is not enough to create this project, so you will need to join forces with other teams.

You will need to team up with students from other courses and cooperate to work on this project (frontend and backend devs, manual testers, sysadmin).




## What are you going to learn?

- Working in a development team
- Agile methodology
- Management tools

# Tasks

## Frontend requirements (This is only for the Frontend course students)

Create the frontend and the user handling backend part of this project.

- There is a page where artworks are listed and users can browse and save artworks (there are some APIs included in the background materials).
- Users can browse, search and saved artworks. Users saved artworks are stored in a database (as well as other user data).
- Artworks have their own page with all their details.
- Users can add tags to their saved artworks (and browse their tags as well).
- When you have to store and image for a saved artwork send it to the Backend students' image server.

## Backend requirements (This is only for the Backend course students)

Create an image server part of this project. Starter template repo: https://github.com/CodecoolGlobal/artproject-backend-starter

- You have to create an image server to save images arriving from the Frontend students' server. (POST request arrive)
- The server saves the posted images in a PostgreSQL database.
- Send back a link where the stored images are available. (GET request arrive)
- You have to create a command line interface, where you can query the uploaded images from the database. You can query by different criteria, for example: size, file extension, category, id. You can also delete the images by id. (optional)
- The backend HTTP endpoints are already implemented in the starter code above. (You can tweak them, but you don't have to start from scratch.)

  Backend API:

  *POST*
  **/api/signup**
  JSON({ email, password }) --->
  <-- 200 / 400, 409
  *POST*
  **/api/login**
  JSON({ email, password }) --->
  <--- 200: JWT / 400, 401
  *POST*
  **/api/artwork**
  FormData({ title, description, imgfile }) + authorization header(JWT) --->
  <--- 200: id / 400, 403
  **GET**
  **/api/artwork**
  authorization header(JWT) --->
  <--- 200: JSON([{ id, title, description, imgurl }]) / 400, 403
  *PATCH*
  **/api/artwork/id**
  JSON({ title, description }) + authorization header(JWT) --->
  <--- 200 / 400,  403
  *DELETE*
  **/api/artwork/id**
  authorization header(JWT) --->
  <--- 200 / 400,  403 (edited)

## Manual Tester requirements ((This is only for the Tester course students)

Create the test part of this project.

- Study the Agile Testing Extension of ISTQB syllabus and acquire all the needed knowledge
- Be the product owner: the whole testing team should take the role together
    - Figure out jira environment: customize the project, create dashboards, administrate a kanban board
    - Write stories and break down to tasks
    - Write acceptance criteria for all stories and tasks
    - Prioritize tasks
    - Lead planning of one week long sprints
- Be the scrum master: one of you takes the role with all related responsibility
    - Organize ceremonies like sprint planning, daily scrum
    - Moderate the communication among team members
    - Be the first point contact to mentors
    - Lead retrospective meetings at the end of sprints
- Bet the tester of scrum team
    - Create detailed test plan
    - Perform the testing activities
        - Write test cases for clear tasks in advance
        - Test with experience based techniques besides planned test cases
        - Use reactive approach, perform the test design afterwards in case of recognised defects
        - Report defects in Jira
    - Organize the versioning of product
    - Provide process management
    - Close testing activity at the end of project
        - Create test summary report with visualisation of gathered data
- Communicate with dev team naturally and neutrally
- Close every sprint with demo
    - Give a presentation at the end of project for every participant

## Sysadmin requirements (This is only for the Sysadmin course students)

**Create the sysadmin part of this project.**

- Create vpc in eu-central-1
- Create a subnet for database  (and other non public stuff)
- Create a Subnet for public facing services (api server, frontend server etc) and a subnet for the non public resources (sql server)
- Launch 2 instances for the frontend servers
- Create a target-group for the frontend servers
- Create a load balancer for the frontend
- Launch instance for the backend server

- Docker files and development environment are set up.
- Deploy the project (preferably with an automated build from github).
- Sysadmins figured out the best infrastructure

## General requirements

## Hints

- "Weeks of coding can save you hours of planning" is a funny quote, meaning you should carefully plan the whole project first before jumping into coding
- You don't need to use the example APIs, you can find an other one you prefer

## Background materials

- Example API#1
- Example API#2
- Example API#3
- Example API#4
- Agile tester