Date: 13-05-2022

Group: S3-CB01

Student: Oleksandr Gurianov

Version: 3.0

Software Architecture

Table of content

How is SOLID guaranteed

Single-responsibility principle

Every class has its single responsibility so that there is no more than one reason for it to change.

Open-closed principle

Every management class is designed in a way that keeps it closed for modification but opened for a later extension.

Liskov substitution principle

Some subclasses use references from their base classes (e.g., a User can be an Administrator or Customer, a pet can be a Cat or Dog, etc.).

Interface segregation principle

This principle is not used at the moment.

Dependency inversion principle

Every class in the business layer is connected via abstraction to the corresponding class in the database layer. Therefore, classes in the business layer are dependent on the classes in the database layer. Meanwhile, every class in the presentation layer is connected via another abstraction to the corresponding class in the business layer. Thus, such dependency allows classes in the presentation layer to connect to multiple classes in the business layer.

Important Design Decisions

Web framework

My choice fell on the open-source, microservice-based Java web framework called Spring Boot because it allows you to quickly build the application jar and execute it without altering the deployment. Furthermore, it automatically generates opinionated dependencies to make building configuration easier.

Front-end library

React, an open-source front-end JavaScript library was my choice because it enables programmers to design massive web apps that can alter data without requiring a page reload. Its primary goal is to be quick, scalable, and easy to use. Furthermore, this library is used by many significant corporations worldwide and has a large user base.

Database

Because H2, a document database, is used to create highly accessible and scalable internet applications, I believe it is the best option for my future web application. Even though I am more experienced with SQL-type databases, this knowledge will aid me in learning H2.

System context (C1)
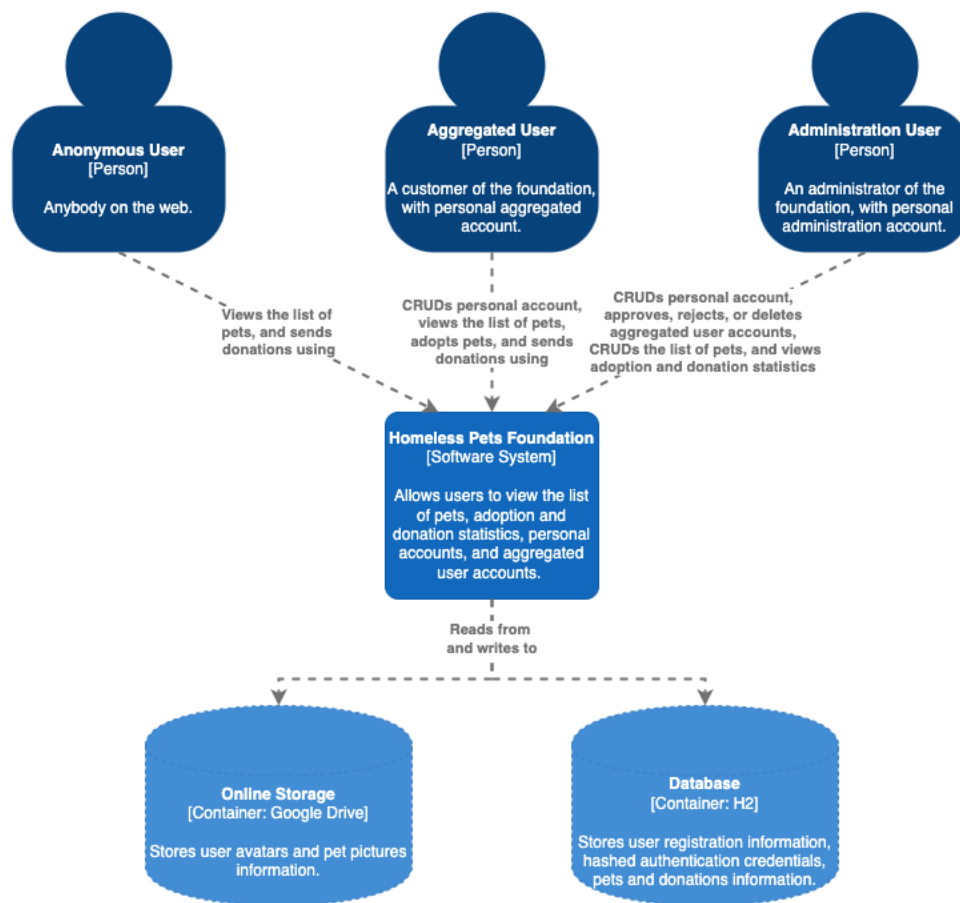
System Context diagram for Homeless Pets Foundation

Explanation:

The diagram represents three user roles: Anonymous User (anybody on the web), Aggregated User (customer), and Administration User (administrator). They use the Homeless Pets Foundation software system which accesses two databases when used: Google Drive to store pictures and H2 to store other data.
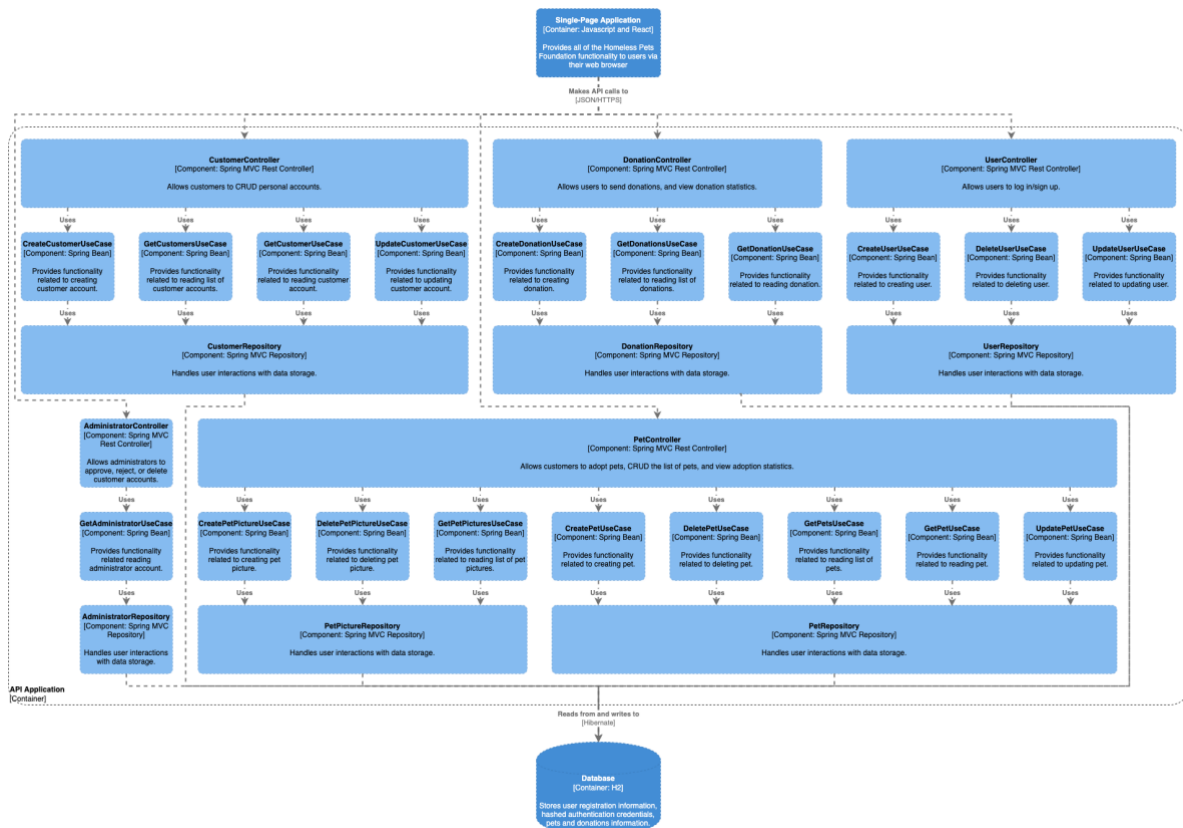
Containers (C2)
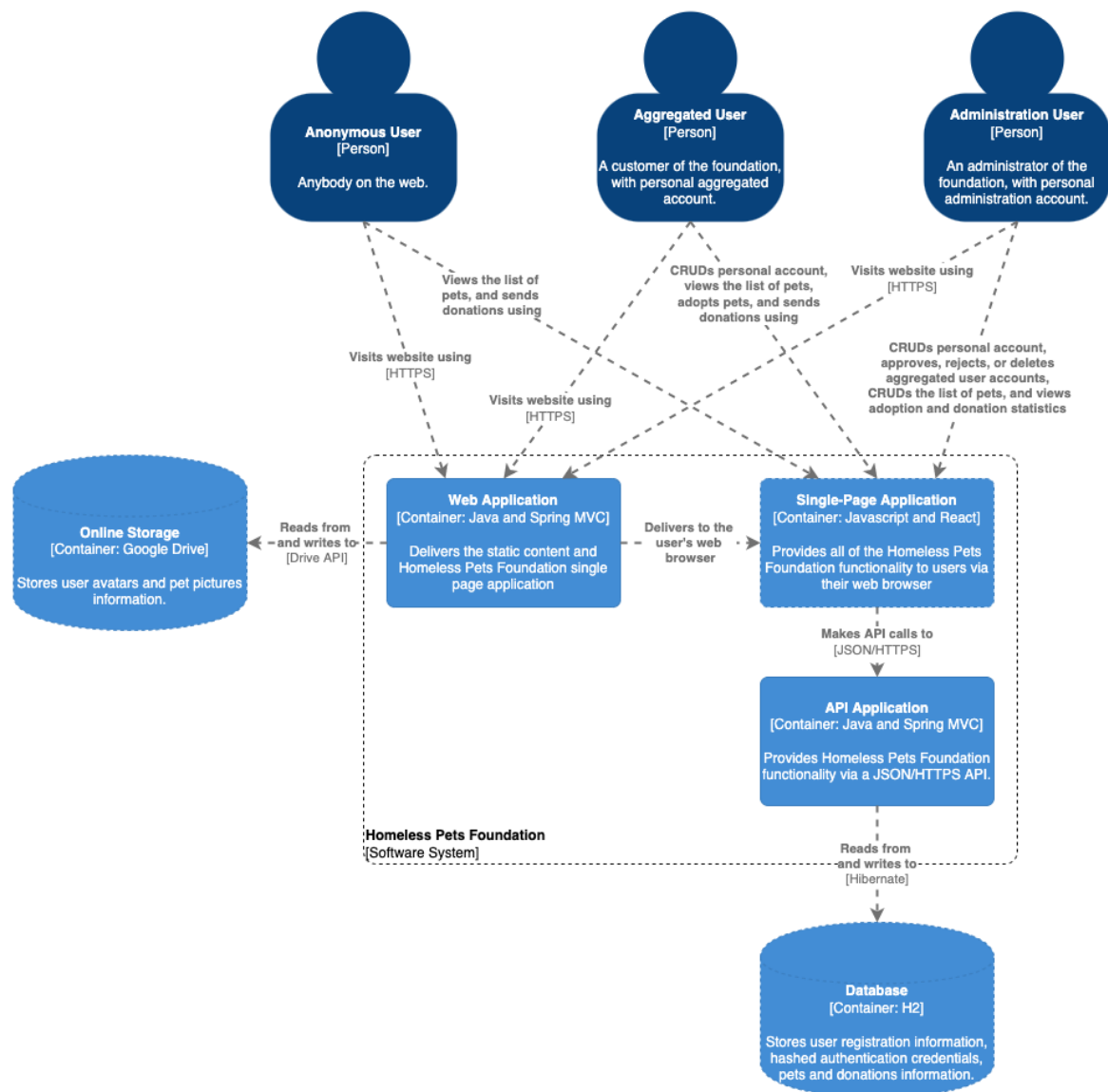
Container diagram for Homeless Pets Foundation

Explanation:

The diagram represents a deeper look into the Homeless Pets Foundation software system. The users use the Web Application to visit the website, and the Single-Page Application to perform various actions. The software system consists of a Web Application that accesses the Google Drive database via Drive API. The Web Application also accesses the H2 database through a Single-Page Application, then an API Application via Hibernate.
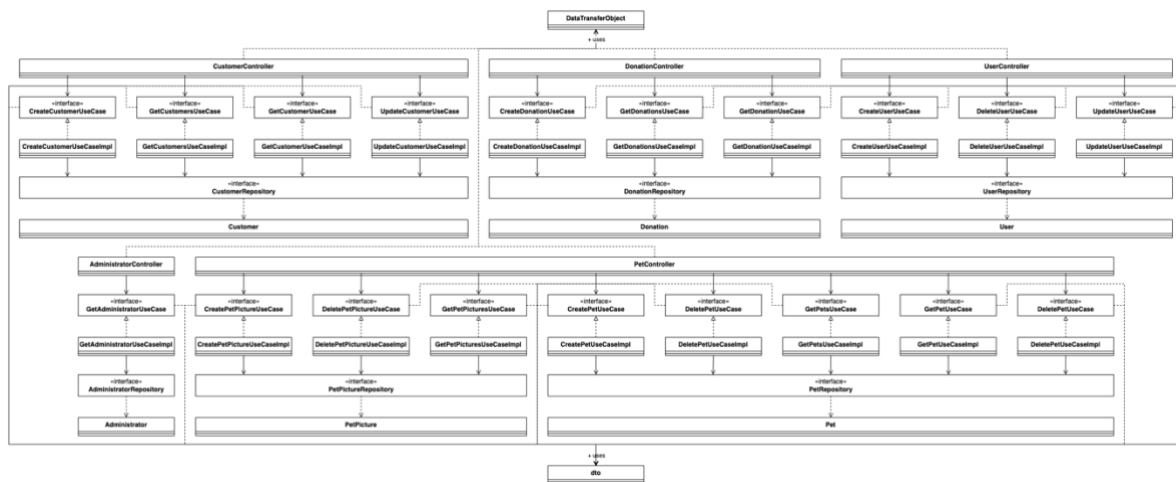
Components (C3)

Component diagram for Homeless Pets Foundation – API Application

Explanation:

The diagram represents a deeper look into an API Application. The users make API calls to it through a Single-Page Application. The API Application container represents a three-layer design that consists of a Controller, Service(s), and Repository(s). Finally, the Repository accesses the H2 database via Hibernate.

Code (C4)

Code diagram for Homeless Pets Foundation – API Application

Explanation:

The diagram represents an even deeper look into an API Application. It reveals a more in-depth look its' five-layered design which consist of a Controller, UseCase interface(s), UseCase implementation(s), Repository(s), and Entity(s). The controller accesses the UseCase implementation(s) through the corresponding interface(s), as well as the corresponding dtos. The Repository accesses the corresponding entity to manipulate data, as well as the Controller accesses the DataTransferObject.