coding {the} architecture

Software Architecture as Code

Simon Brown
@simonbrown

“…the architecture diagrams don’t match the code”

Kristijan | Криштиџн
@Krishtidzn

Follow

Any recommendations for software for drawing software architecture but not MS Visio?

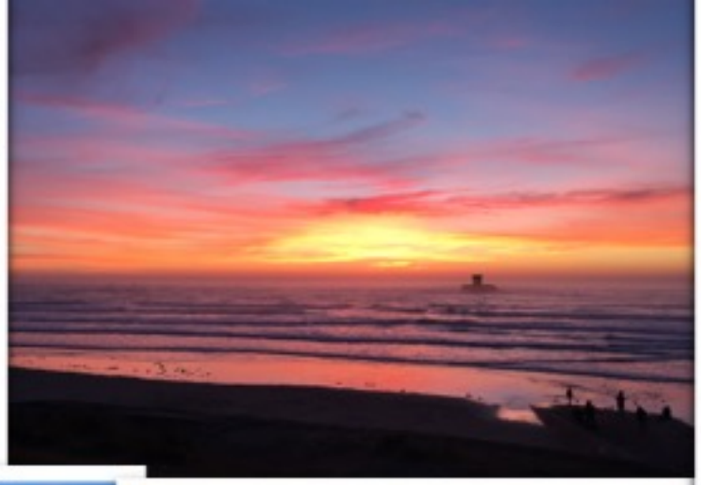
11:11 AM - 16 Apr 2015

Greenland
Iceland
Norwegian Sea
Sweden
Finland
Norway
Denmark
United Kingdom
Ireland
Poland
Germany
Belarus
Jersey
Austria
France
Romania
Italy
Spain
Portugal
Greece
Tunisia
Morocco
Algeria
Libya
Western
North Atlantic Ocean
NL
QC
NB
PE
NS
ME
NH
MA
NY
PA
WV
DE
VA
NC
SC
FL

Jersey

Guernsey

Jersey





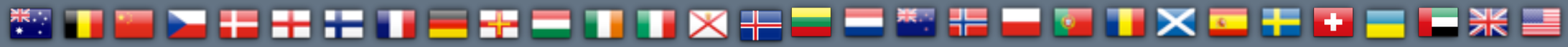







I help software teams understand

software architecture, technical leadership and the balance with agility



Software architecture needs to be more accessible

coding {the} architecture

Software Architecture for Developers

Technical leadership by **coding**, coaching, collaboration, *architecture sketching* and just enough up front design

Simon Brown

Leanpub

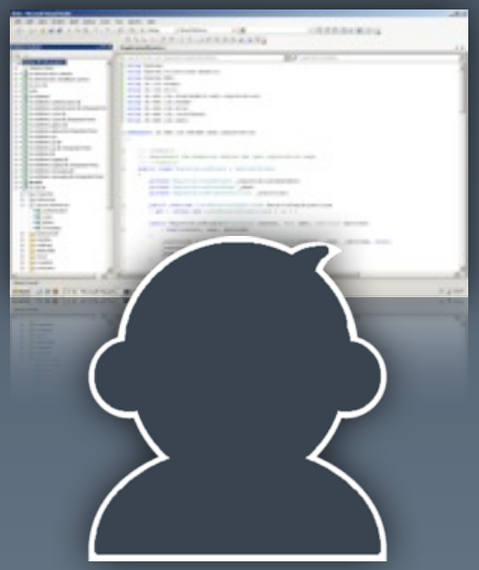
coding {the} architecture

The Art of Visualising Software Architecture

Communicating software architecture with *sketches*, diagrams and the C4 model

Simon Brown

Free!



I code too

The intersection between

software architecture

and

code

The tension between software architecture and code

How do we

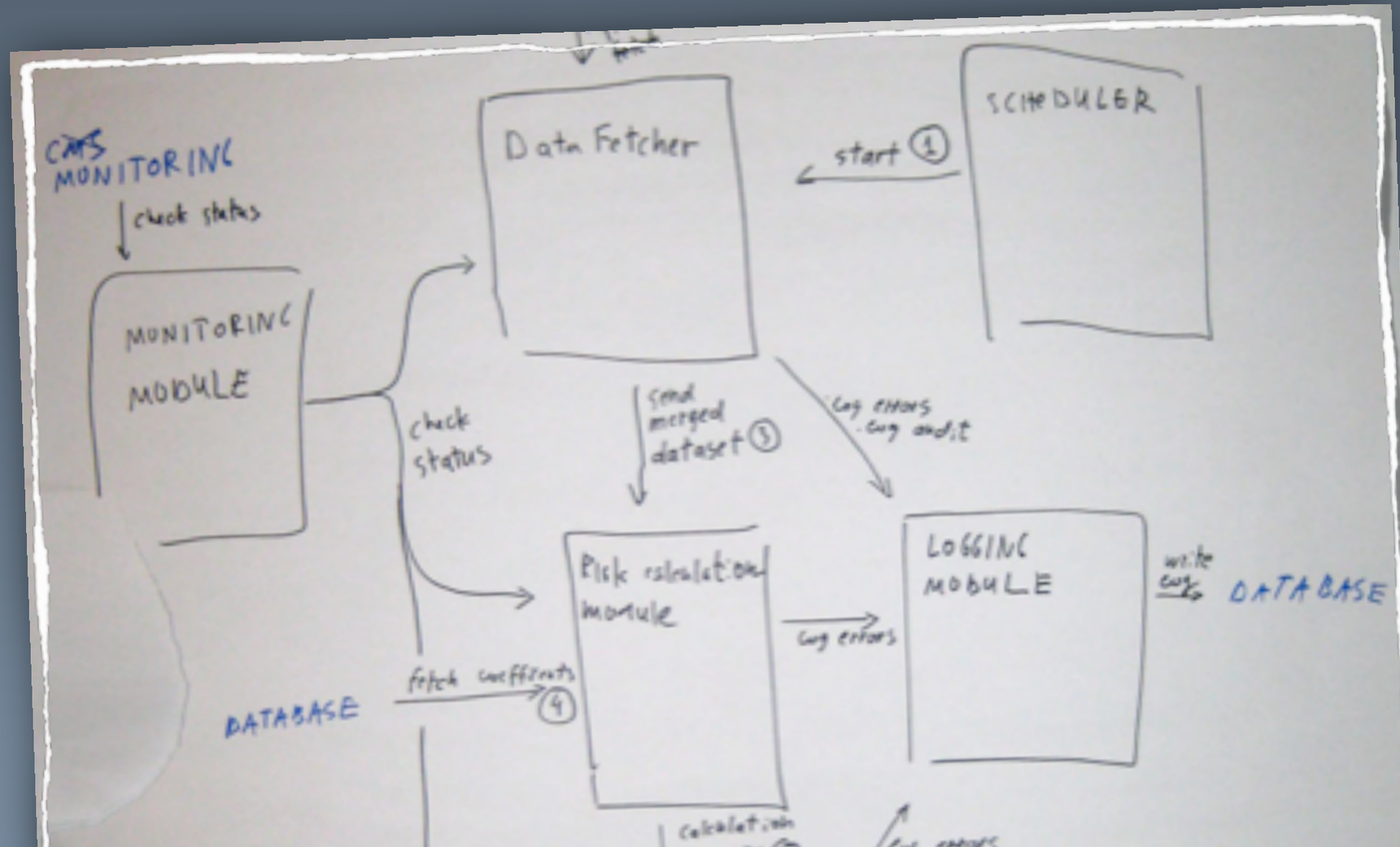
communicate

software architecture?

Abstraction

is about reducing detail

rather than creating a different representation



Abstractions help us

reason about

a big and/or complex
software system

Software architecture deals with abstraction, with decomposition and composition, with style and esthetics.

To describe a software architecture, we use a model composed of multiple views or perspectives.

Architectural Blueprints—The “4+1” View Model of Software Architecture

by Philippe Kruchten

The description of an architecture—the decisions made—can be organized around these four views, and then illustrated by a few selected *use cases*, or *scenarios* which become a fifth view. The architecture is in fact partially evolved from these scenarios as we will see later.

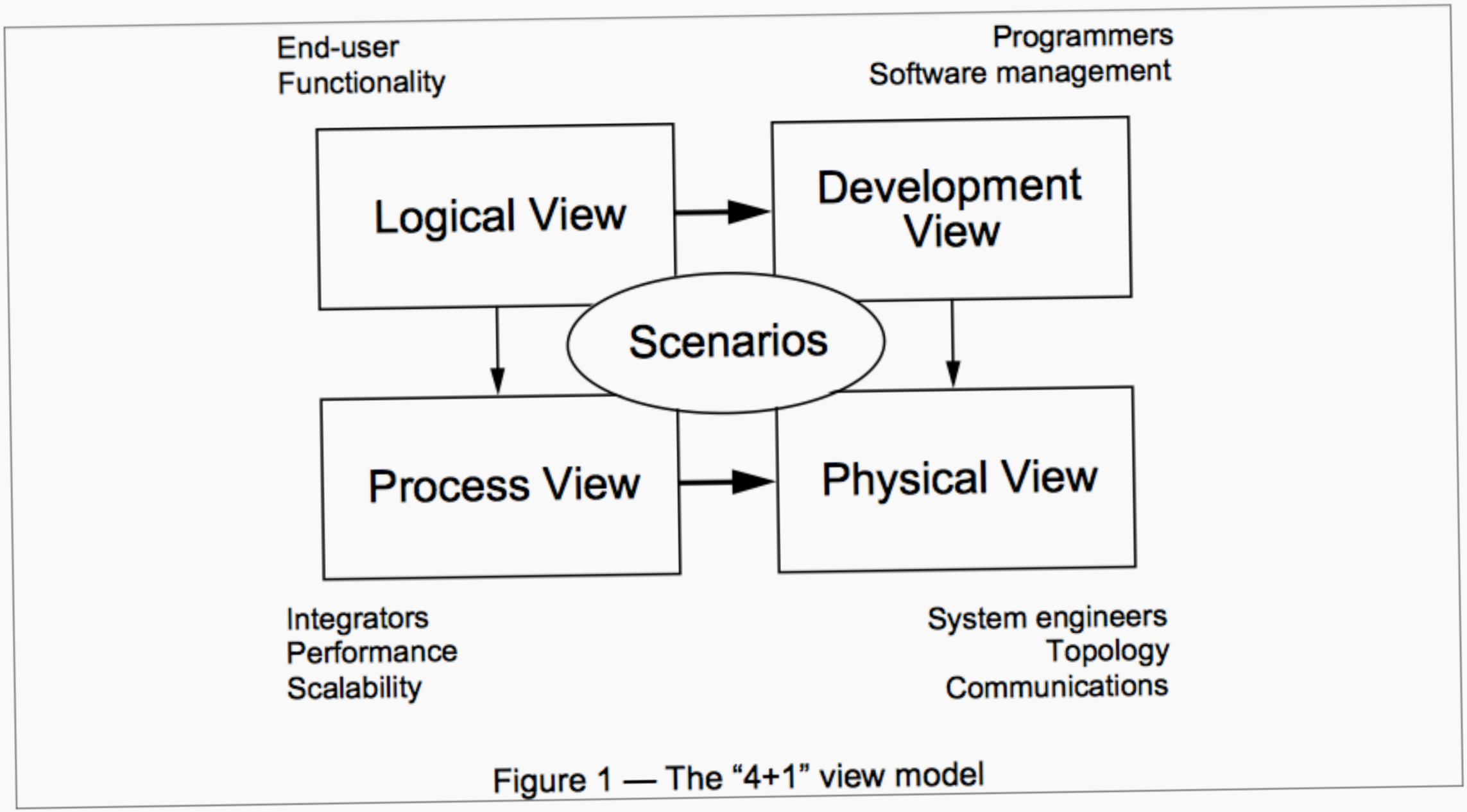


Figure 1 — The "4+1" view model

We apply Perry & Wolf's equation independently on each view, i.e., for each view we define the set of elements to use (components, containers, and connectors) , we capture the forms and patterns that work, and we capture the rationale and constraints, connecting the architecture to some of the requirements.

Do the **names**
of those views make sense?

Conceptual vs Logical

Process vs Functional

Development vs Physical

Development vs Implementation

Physical vs Implementation

Physical vs Deployment

Logical and development views are often separated

Would we

code

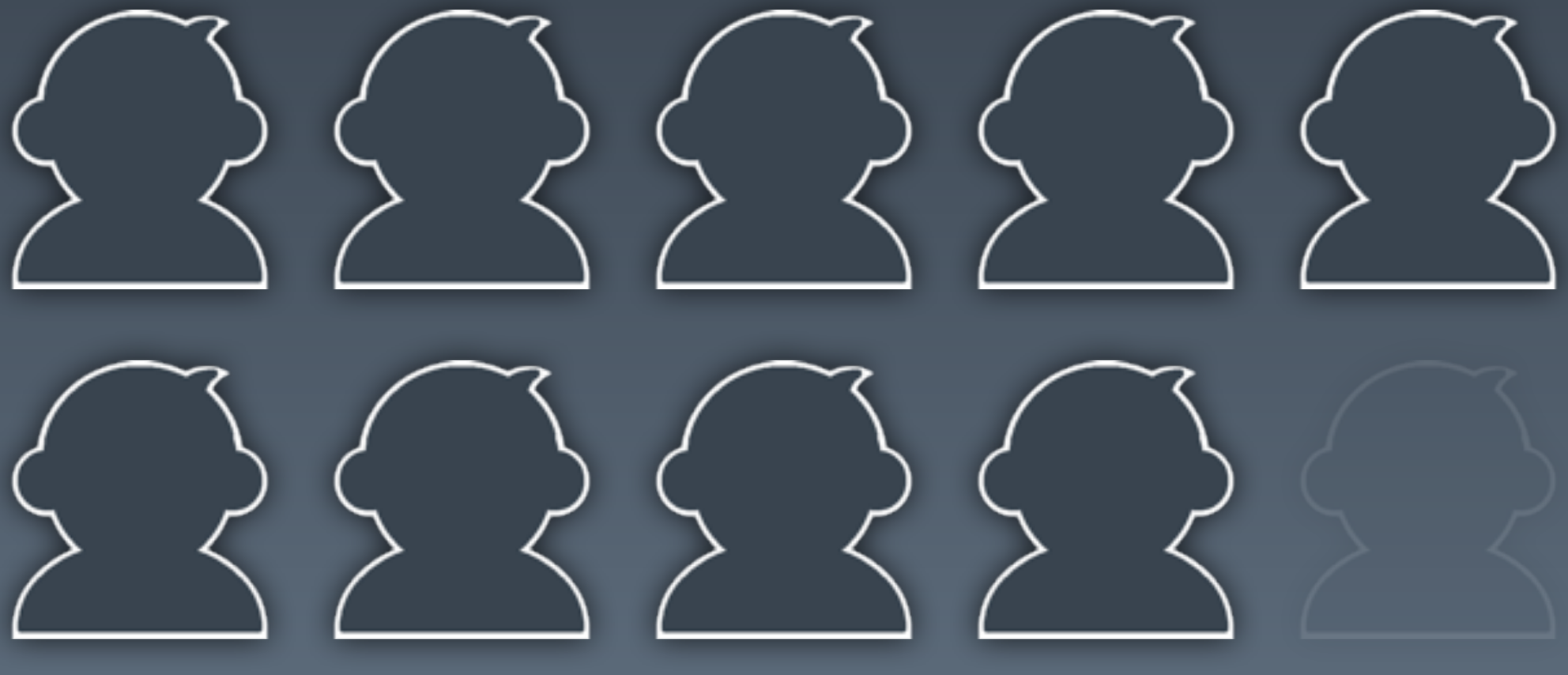
it that way?

Did we

code

it that way?

As an industry, we lack a common vocabulary with which to think about, describe and communicate software architecture

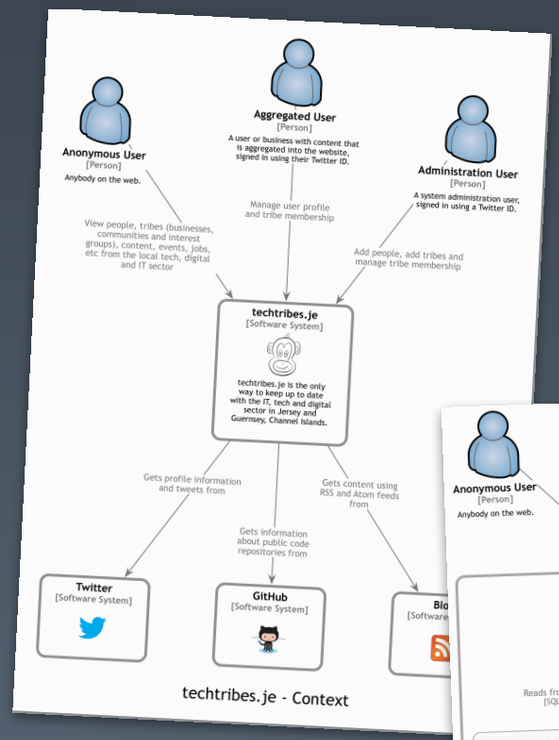


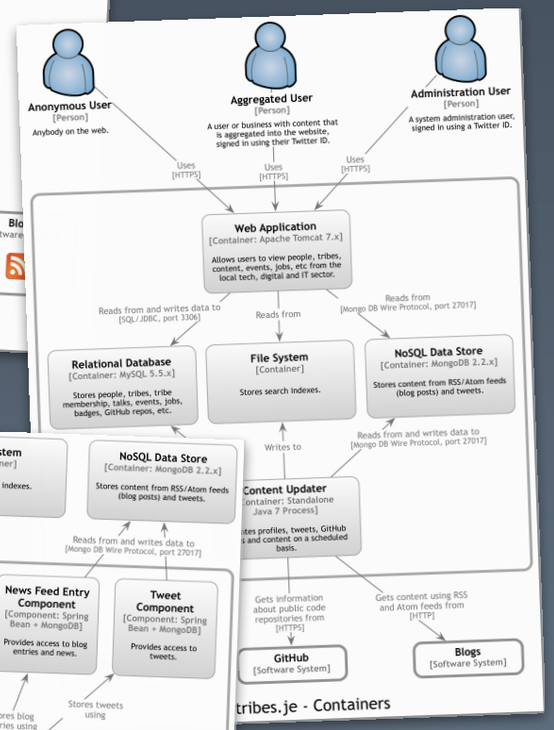
9 out of 10 people don't use UML

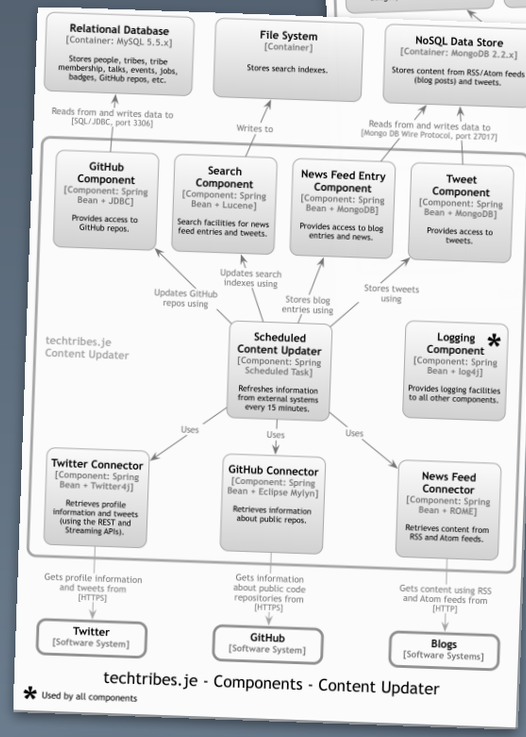
(in my experience)

A common set of abstractions

is more important than
a common notation

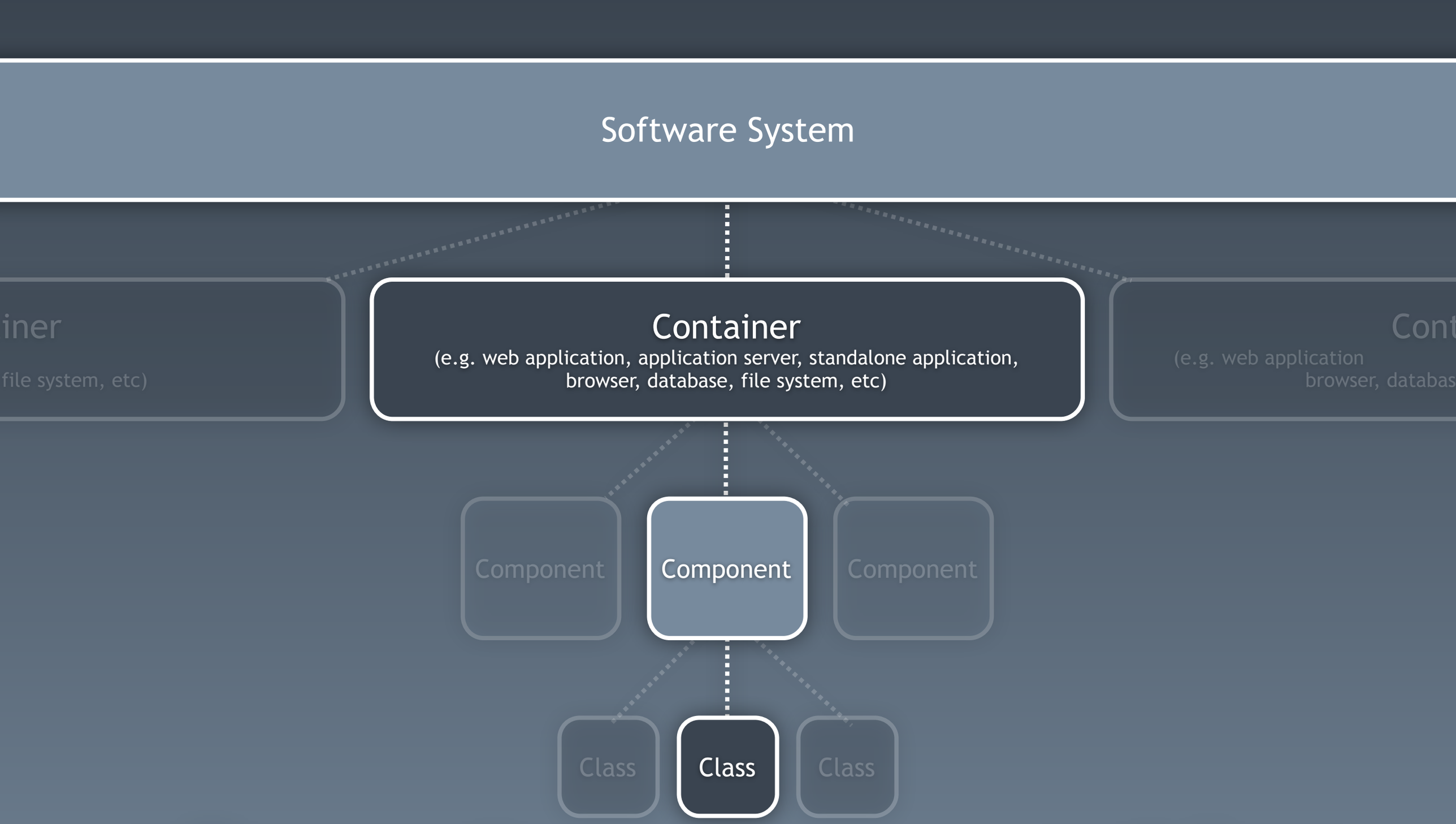






Diagrams are maps

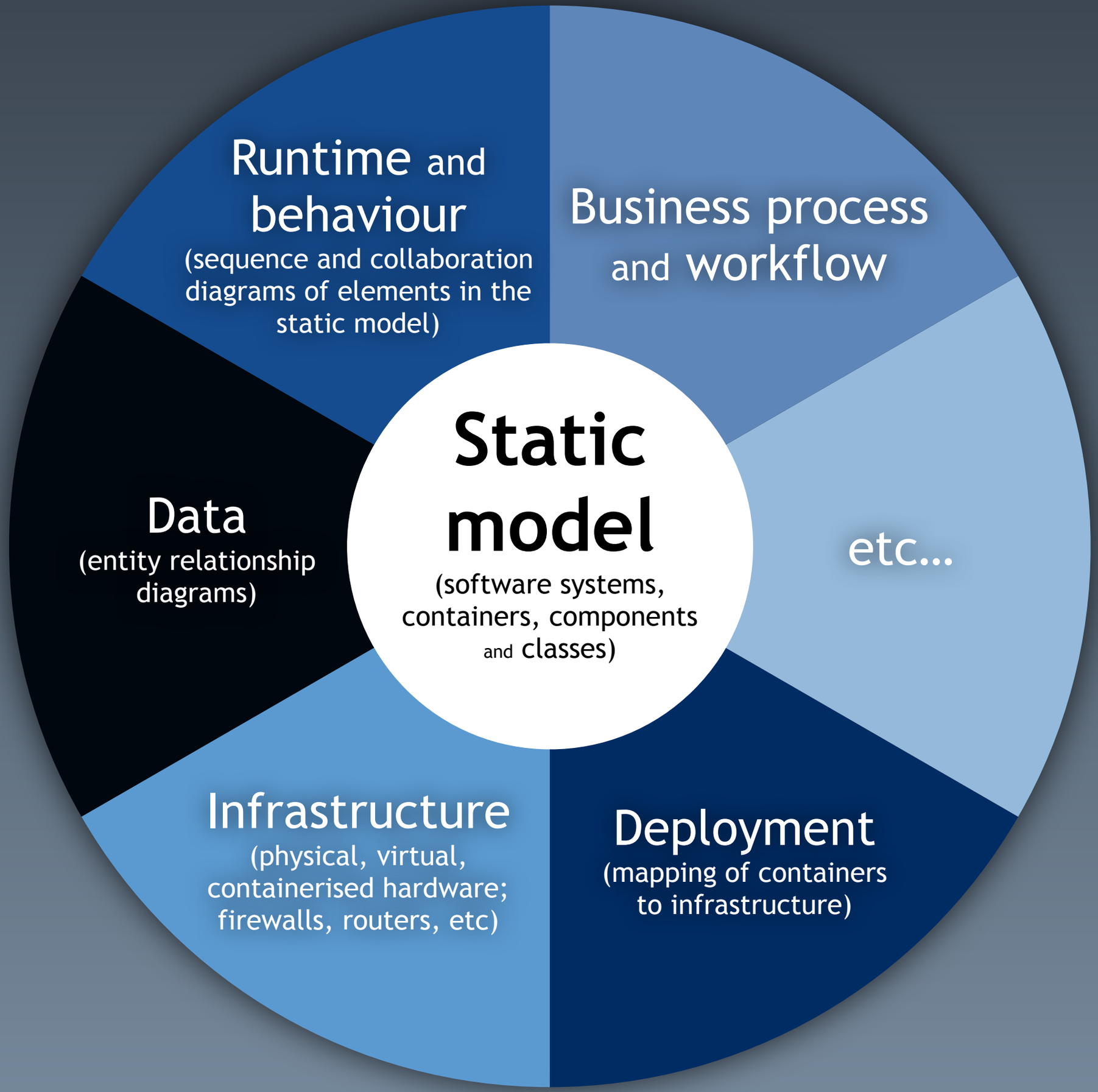
that help a team navigate a complex codebase



A **software system** is made up of one or more **containers**,

each of which contains one or more **components**,

which in turn are implemented by one or more **classes**.


Runtime and behaviour
(sequence and collaboration diagrams of elements in the static model)
Business process and workflow
Static model
(software systems, containers, components and classes)
etc...
Data
(entity relationship diagrams)
Infrastructure
(physical, virtual, containerised hardware; firewalls, routers, etc)
Deployment
(mapping of containers to infrastructure)

C4++

Enterprise context

User interface mockups and wireframes

Domain model

Sequence and collaboration diagrams

Business process and workflow models

Infrastructure model

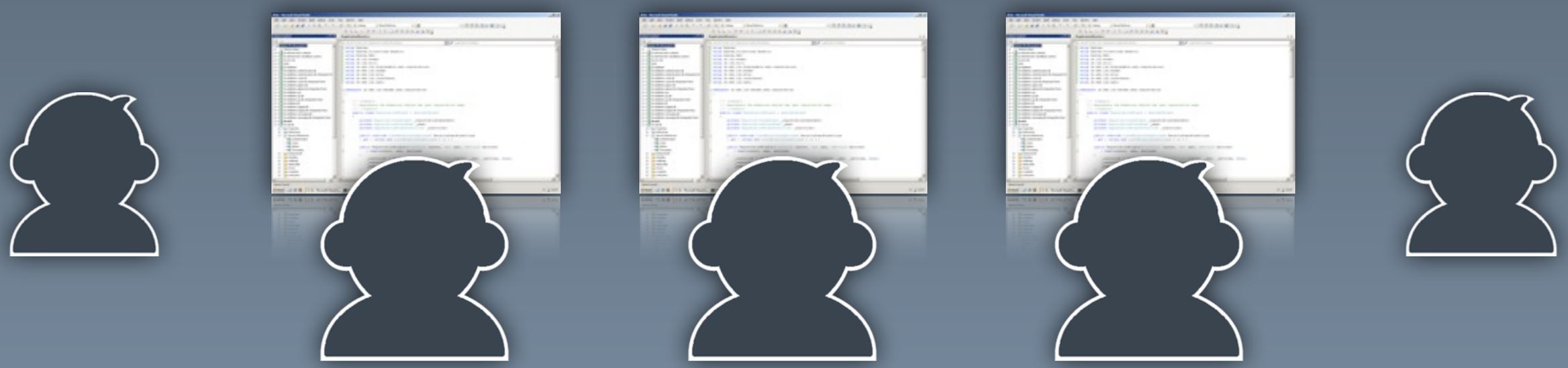
Deployment model

...

Software developers are the most important

stakeholders

of software architecture

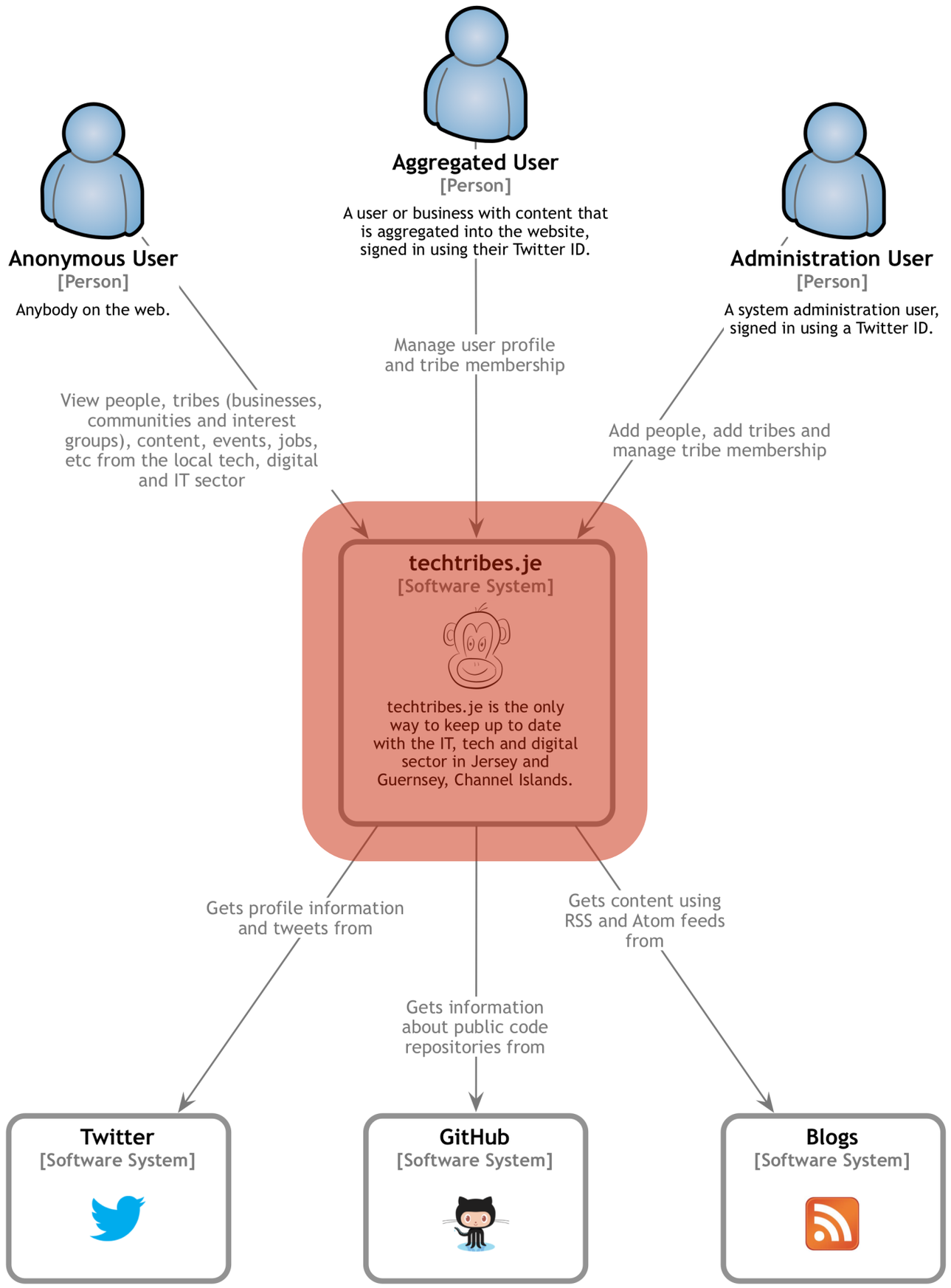


Context diagram
(level 1)

Container diagram
(level 2)

Component diagram
(level 3)

Class diagram
(level 4)



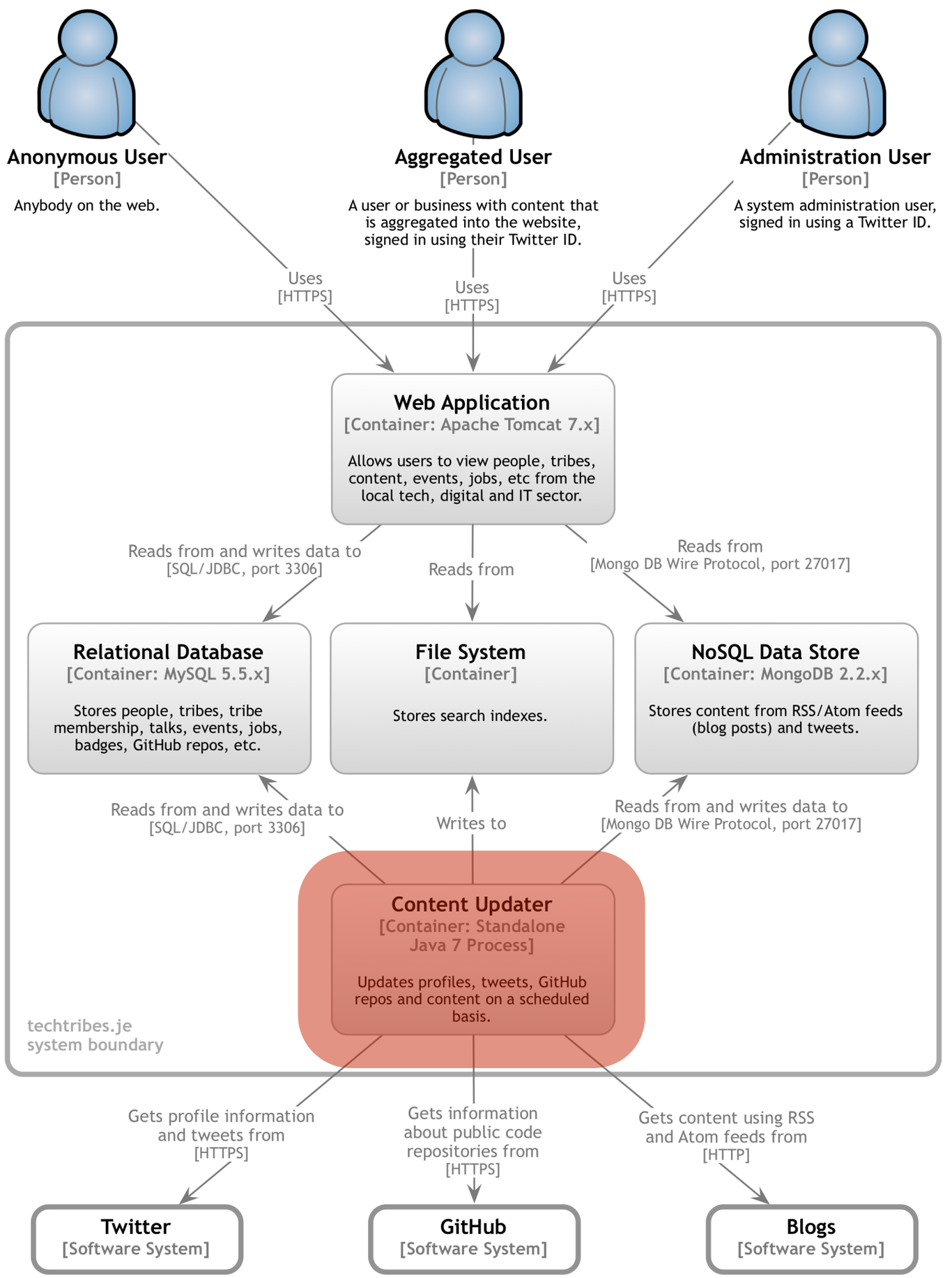
techtribes.je - Context

Context diagram
(level 1)

Container diagram
(level 2)

Component diagram
(level 3)

Class diagram
(level 4)



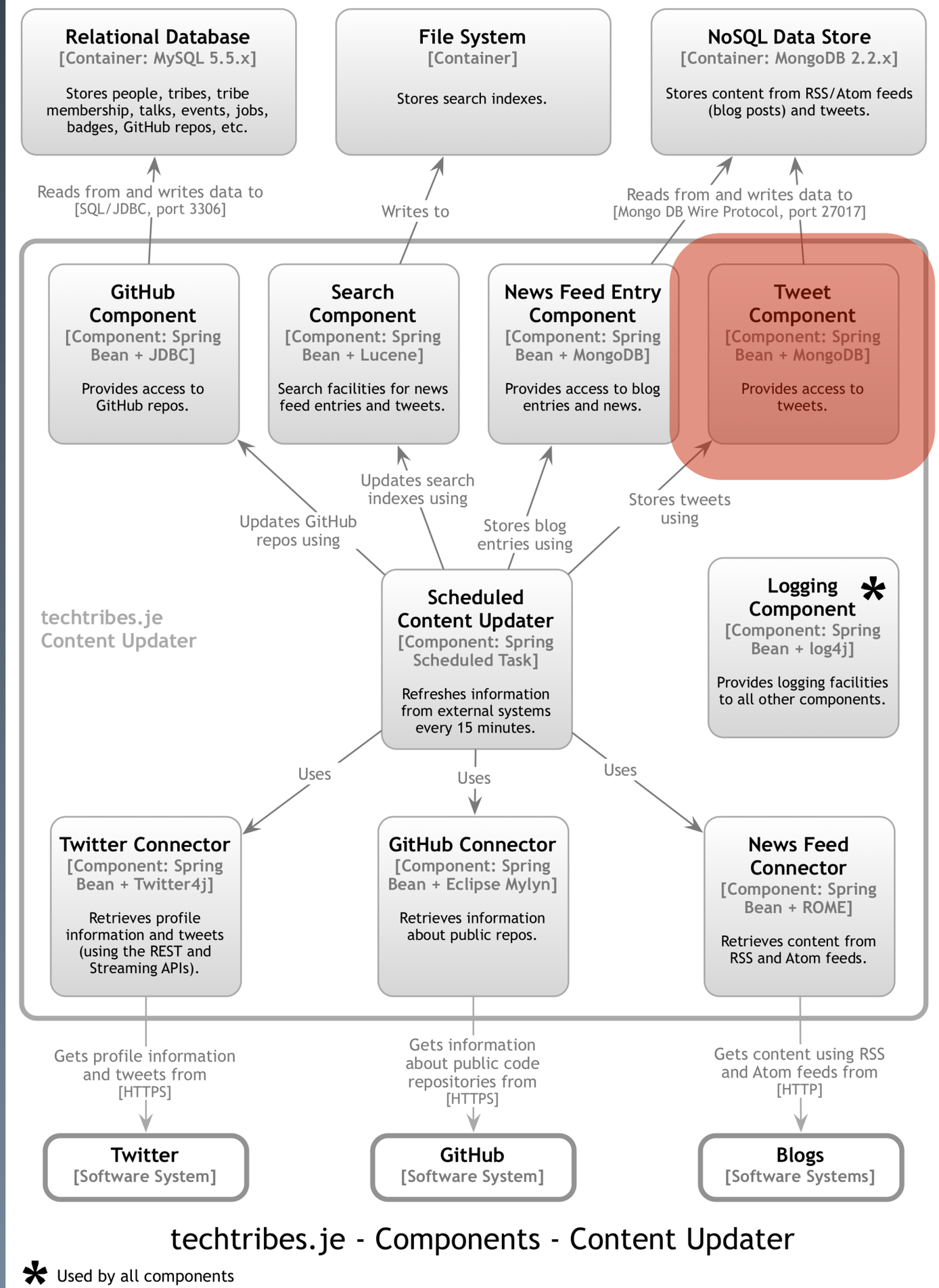
techtribes.je - Containers

Context diagram
(level 1)

Container diagram
(level 2)

Component diagram
(level 3)

Class diagram
(level 4)



techtribes.je - Components - Content Updater

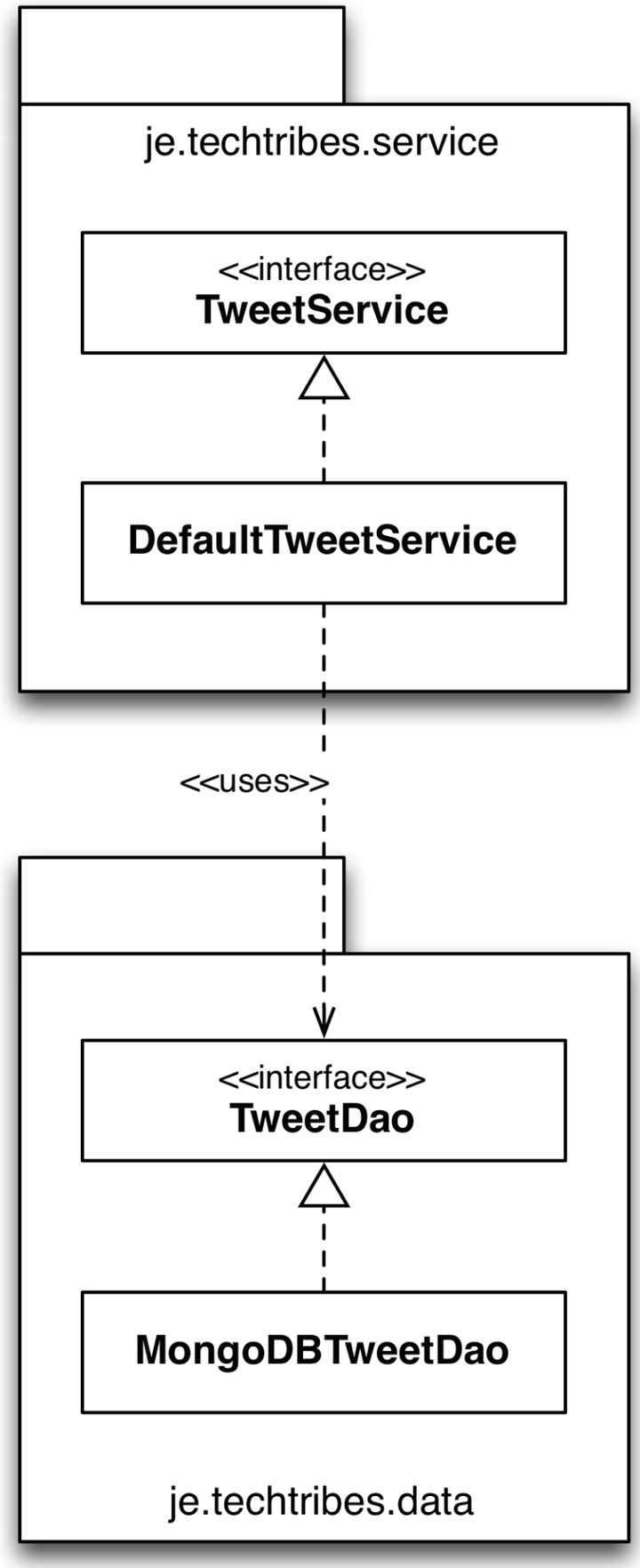
* Used by all components

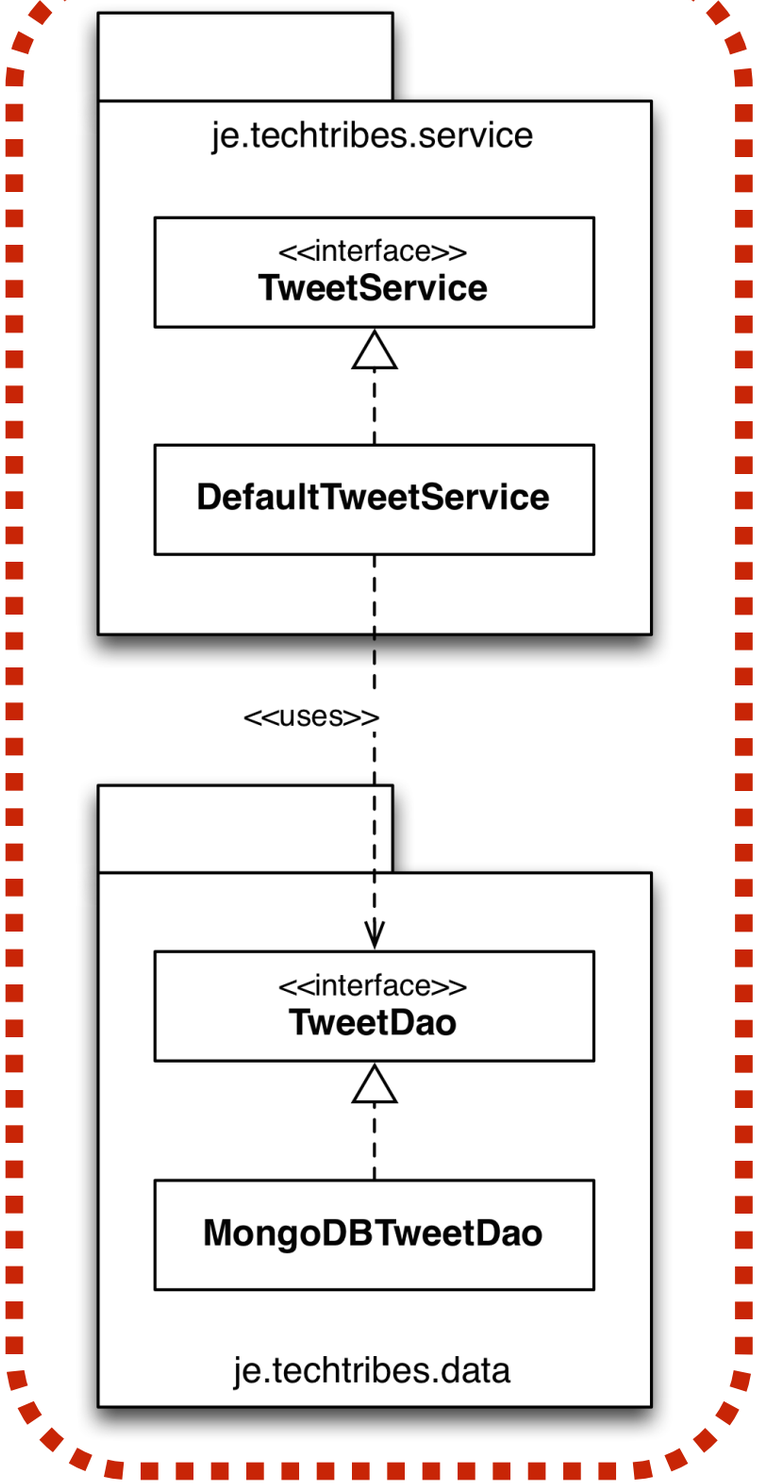
Context diagram
(level 1)

Container diagram
(level 2)

Component diagram
(level 3)

Class diagram
(level 4)





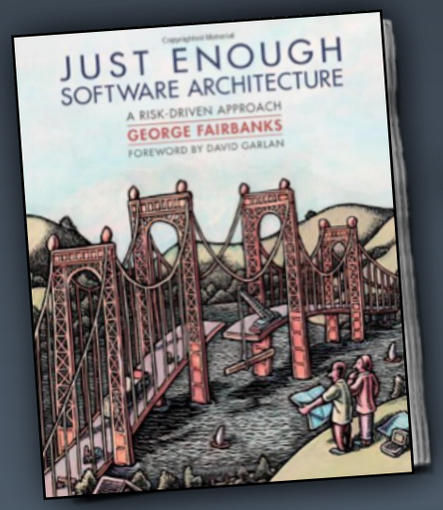
The
“Tweet Component”
doesn’t exist as a
single thing …
it’s a combination of
interfaces and classes
across a layered
architecture

Where’s my component?

Does your code reflect the

abstractions

that you think about?



“the model-code gap”

Model-code gap. Your architecture models and your source code will not show the same things. The difference between them is the *model-code gap*. Your architecture models include some abstract concepts, like components, that your programming language does not, but could. Beyond that, architecture models include intensional elements, like design decisions and constraints, that cannot be expressed in procedural source code at all.

Consequently, the relationship between the architecture model and source code is complicated. It is mostly a refinement relationship, where the extensional elements in the architecture model are refined into extensional elements in source code. This is shown in Figure 10.3. However, intensional elements are not refined into corresponding elements in source code.

Upon learning about the model-code gap, your first instinct may be to avoid it. But reflecting on the origins of the gap gives little hope of a general solution in the short term: architecture models help you reason about complexity and scale because they are abstract and intensional; source code executes on machines because it is concrete and extensional.

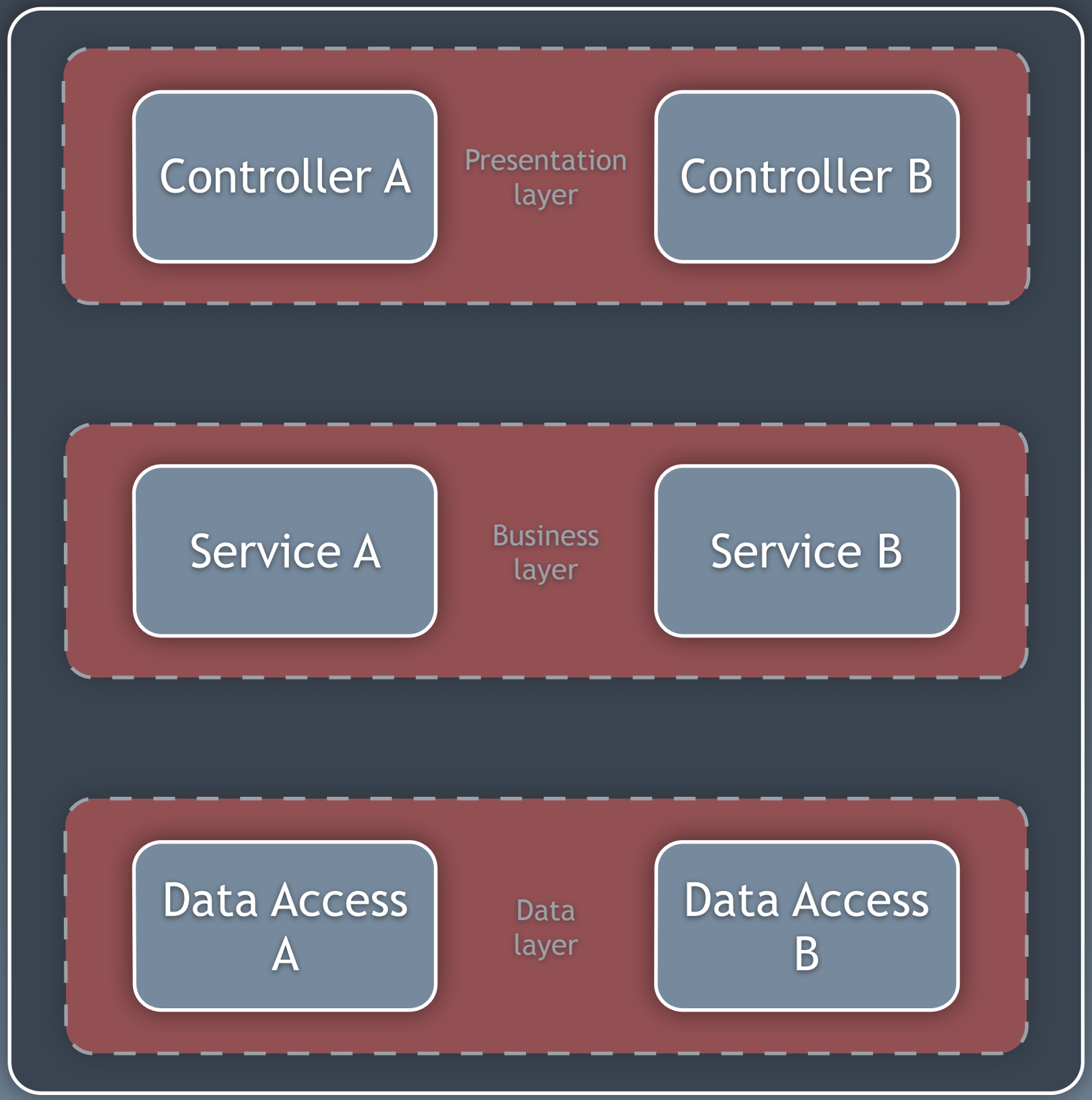
Organisation of code vs the architectural views

We often think

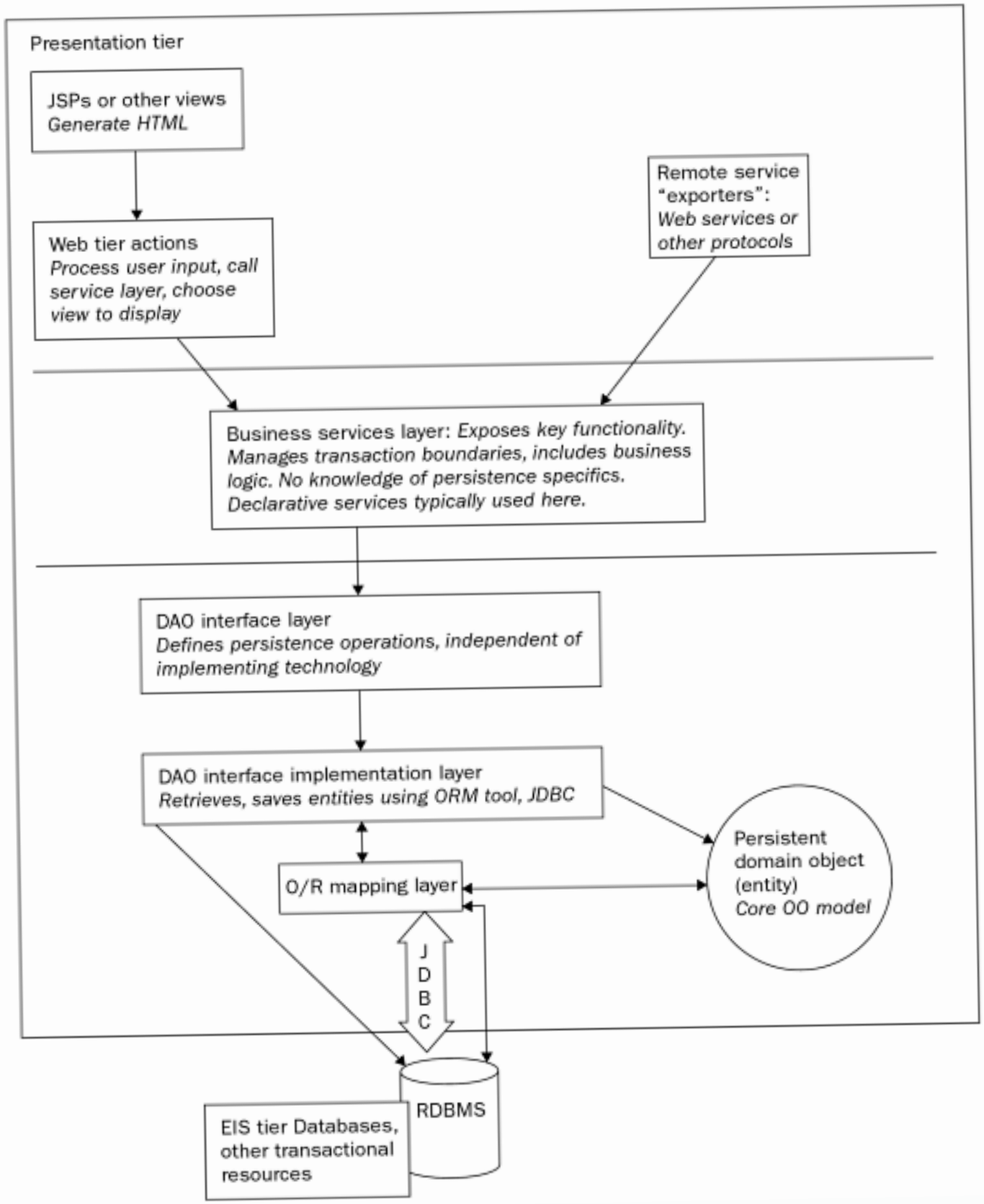
in components

but write classes

(usually in layers)



Package by layer (horizontal slicing)


Presentation tier
JSPs or other views
Generate HTML
Web tier actions
Process user input, call service layer, choose view to display
Remote service "exporters":
Web services or other protocols
Business services layer: Exposes key functionality. Manages transaction boundaries, includes business logic. No knowledge of persistence specifics. Declarative services typically used here.
DAO interface layer
Defines persistence operations, independent of implementing technology
DAO interface implementation layer
Retrieves, saves entities using ORM tool, JDBC
O/R mapping layer
Persistent domain object (entity)
Core OO model
J
D
B
C
RDBMS
EIS tier Databases, other transactional resources

Let's summarize each layer and its responsibilities, beginning closest to the database or other enterprise resources:

- **Presentation layer:** This is most likely to be a web tier. This layer should be as thin as possible. It should be possible to have alternative presentation layers — such as a web tier or remote web services facade — on a single, well-designed middle tier.
- **Business services layer:** This is responsible for transactional boundaries and providing an entry point for operations on the system as a whole. This layer should have no knowledge of presentation concerns, and should be reusable.
- **DAO interface layer:** This is a layer of interfaces *independent of any data access technology* that is used to find and persist persistent objects. This layer effectively consists of *Strategy* interfaces for the Business services layer. This layer should not contain business logic. Implementations of these interfaces will normally use an O/R mapping technology or Spring's JDBC abstraction.
- **Persistent domain objects:** These model real objects or concepts such as a bank account.
- **Databases and legacy systems:** By far the most common case is a single RDBMS. However, there may be multiple databases, or a mix of databases and other transactional or non-transactional legacy systems or other enterprise resources. The same fundamental architecture is applicable in either case. This is often referred to as the *EIS* (*Enterprise Information System*) tier.

In a J2EE application, all layers except the EIS tier will run in the application server or web container. Domain objects will typically be passed up to the presentation layer, which will display data they contain, *but not modify them*, which will occur only within the transactional boundaries defined by the business services layer. Thus there is no need for distinct Transfer Objects, as used in traditional J2EE architecture.

In the following sections we'll discuss each of these layers in turn, beginning closest to the database.

Spring aims to decouple architectural layers, so that each layer can be modified as far as possible without impacting other layers. No layer is aware of the concerns of the layer above; as far as possible, dependency is purely on the layer immediately below. Dependency between layers is normally in the form of interfaces, ensuring that coupling is as loose as possible.

Java - Eclipse Platform - /Users/Shared/Projects

Java | Java EE

Package Explorer | JUnit

- springapp
  - src
    - springapp.domain
      - Product.java
    - springapp.repository
      - JdbcProductDao.java
      - ProductDao.java
    - springapp.service
      - PriceIncrease.java
      - PriceIncreaseValidator.java
      - ProductManager.java
      - SimpleProductManager.java
    - springapp.web
  - test
  - JRE System Library [JVM 1.5.0 (MacOS X Default)]
  - Referenced Libraries
  - db
  - war
    - WEB-INF
      - classes
        - springapp
        - jdbc.properties
        - messages.properties
      - jsp
        - hello.jsp
        - include.jsp

1 Answer

active oldest votes

8

If you are using maven, it's best to follow the standard maven project layout. You can get maven to generate this structure for you by doing,

```
mvn archetype:generate
```

and select spring-mvc-jpa-archetype from the list of choices

This will give you a package structure like,

```
├── pom.xml
└── src
    ├── main
    │   ├── java
    │   │   └── mygroup
    │   │       ├── controller
    │   │       │   ├── HomeController.java
    │   │       │   └── PersonController.java
    │   │       ├── dao
    │   │       │   └── PersonDao.java
    │   │       └── model
    │   │           └── Person.java
    │   ├── resources
    │   │   ├── db.properties
    │   │   ├── log4j.xml
    │   │   └── META-INF
    │   │       └── persistence.xml
    │   └── webapp
    │       ├── index.html
    │       ├── META-INF
    │       │   ├── context.xml
    │       │   └── MANIFEST.MF
    │       ├── resources
    │       │   └── css
    │       │       └── screen.css
    │       └── WEB-INF
    │           ├── spring
    │           │   ├── app
    │           │   │   ├── controllers.xml
    │           │   │   └── servlet-context.xml
    │           │   ├── db.xml
    │           │   └── root-context.xml
    │           ├── views
```


Related

2 How do I use a custom authentication mechanism for a Java web application with Spring Security?

0 Is there a sample web project of integrating Spring-MVC and Spring-Data-JPA?

1 Reusing DAOs in another web application

0 Java Web Application Warming

3 Is a parallel Spring-MVC application possible with a non-spring web app?

2 Adding a web interface (Spring MVC) to existing Java application

0 package strucutre for Spring MVC project with multiple sub projects using maven

0 How to add a setup

MvcMovie - Microsoft Visual Studio Expres...

Quick Launch (Ctrl+Q)

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST WINDOW HELP

Internet Explorer | Debug

HomeController.cs

MvcMovie.Controllers.HomeCont | Index()

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MvcMovie.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            ViewBag.Message = "Modify this templ

            return View();
        }

        public ActionResult About()
        {
            ViewBag.Message = "Your app descript

            return View();
        }
```

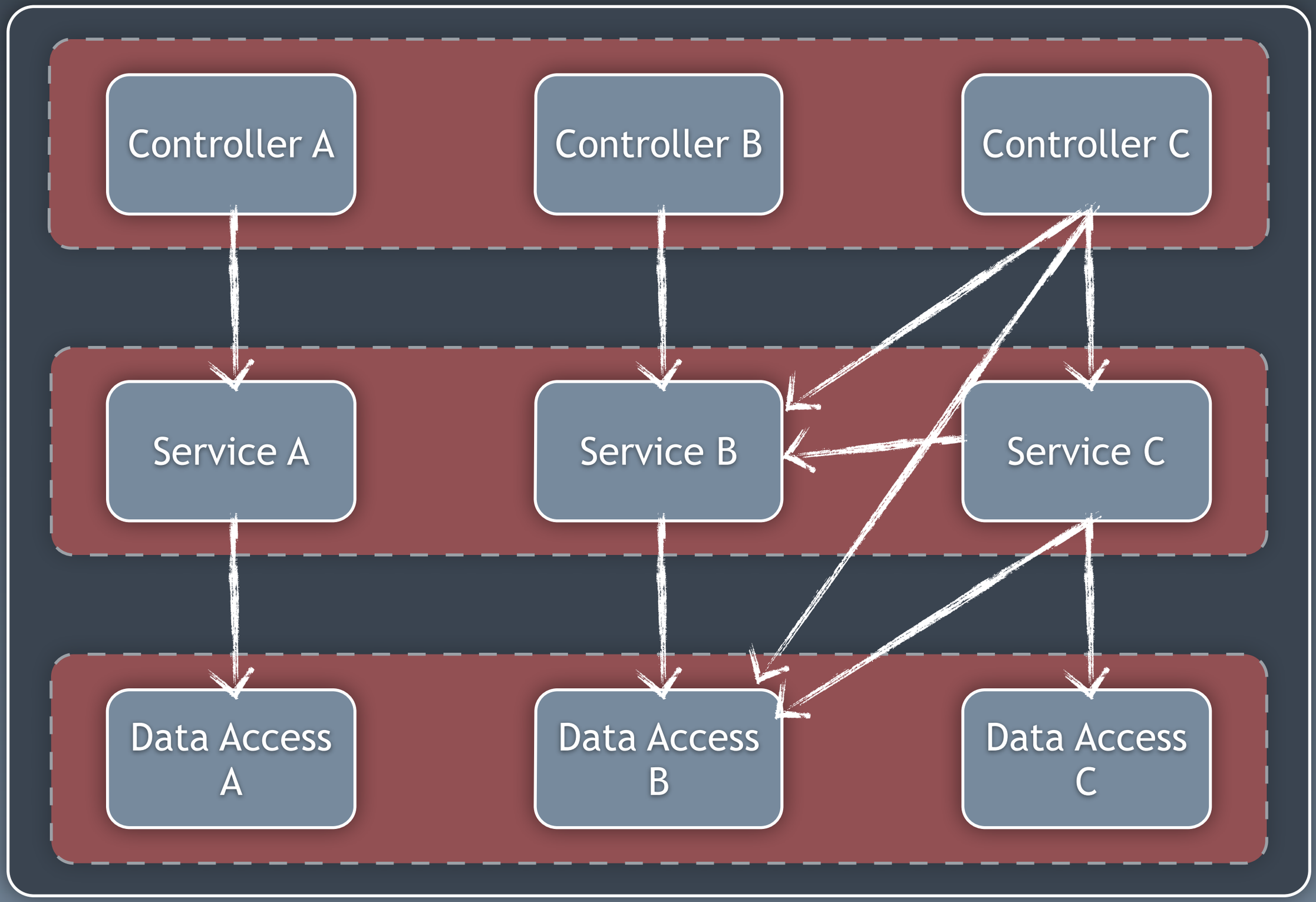
100 %

Solution Explorer

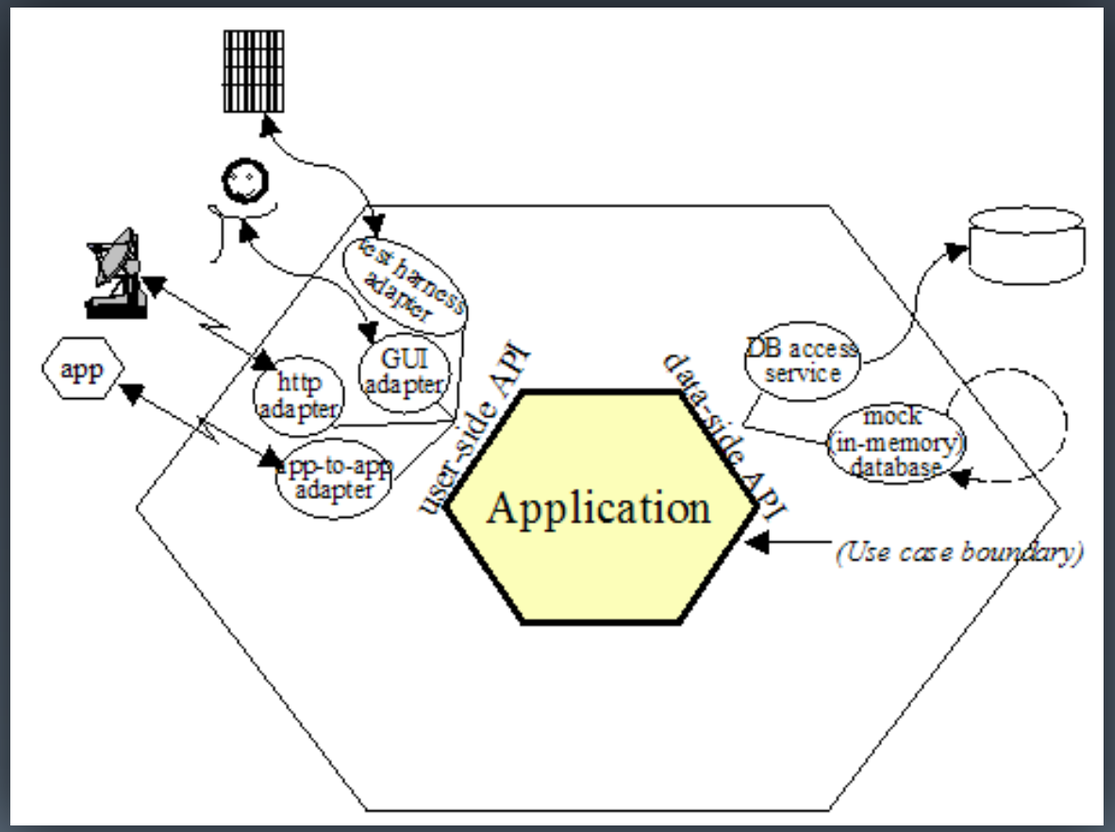
Search Solution Explorer (Ctrl+;)

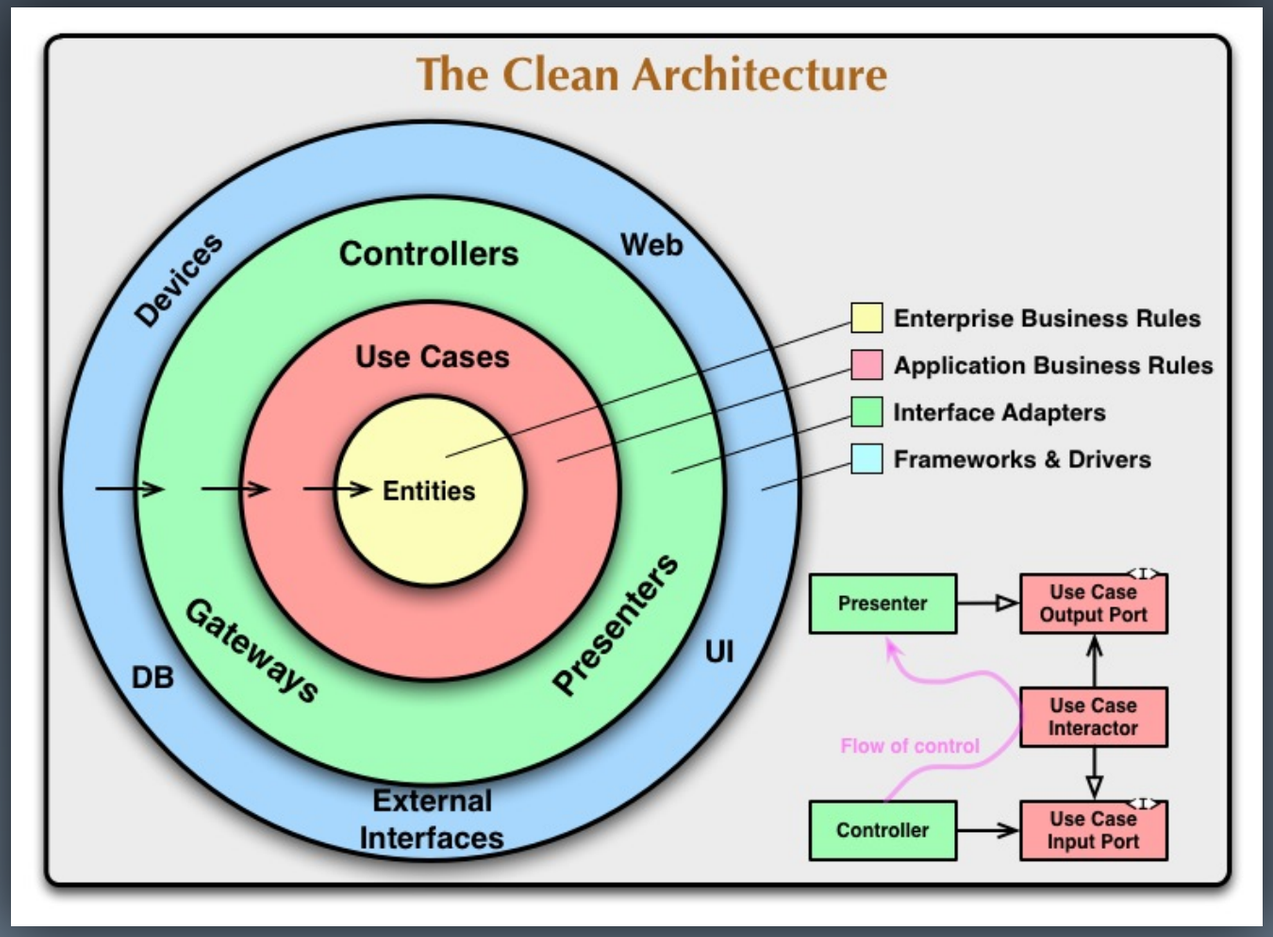
- Solution 'MvcMovie' (1 project)
  - MvcMovie
    - Properties
    - References
    - App_Data
    - App_Start
    - Content
    - Controllers
    - Filters
    - Images
    - Models
    - Scripts
    - Views
    - favicon.ico
    - Global.asax
    - packages.config
    - Web.config

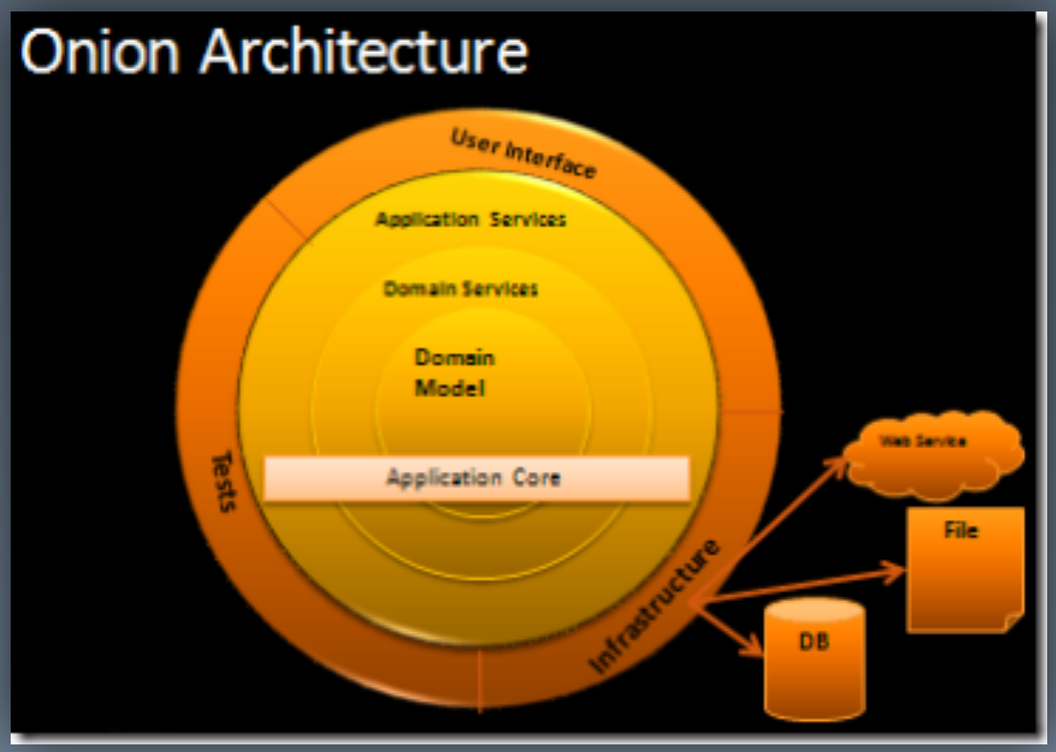
Ready | Ln 1 | Col 1 | Ch 1 | INS



Package by layer (horizontal slicing)







Hexagons and onions

Should layers be

considered

harmful?

Are layers significant

structural elements

or just an

implementation detail?

LINKS

Blogroll
Subscribe via RSS

AUTHORS

Uncle Bob
Paul Pagel
Micah Martin
Eric Smith
Doug Bradbury
Colin Jones
Mike Jansen
Jim Suchy
Craig Demyanovich
Dariusz Pasciak
Kevin Buchanan
Eric Meyer
Myles Megyesi
Dave Moore
Chris Peak
Patrick Gombert
Margaret Pagel
Steve Kim
Sandro Padin
Kevin Liddle
Malcolm Newsome

Screaming Architecture

Uncle Bob → 30 Sep 2011 + Architecture

Imagine that you are looking at the blueprints of a building. This document, prepared by an architect, tells you the plans for the building. What do these plans tell you?

If the plans you are looking at are for a single family residence, then you'll likely see a front entrance, a foyer leading to a living room and perhaps a dining room. There'll likely be a kitchen a short distance away, close to the dining room. Perhaps a dinette area next to the kitchen, and probably a family room close to that. As you looked at those plans, there'd be no question that you were looking at a *house*. The architecture would *scream*: **house**.

Or if you were looking at the architecture of a library, you'd likely see a grand entrance, an area for check-in-out clerks, reading areas, small conference rooms, and gallery after gallery capable of holding bookshelves for all the books in the library. That architecture would *scream*: **Library**.

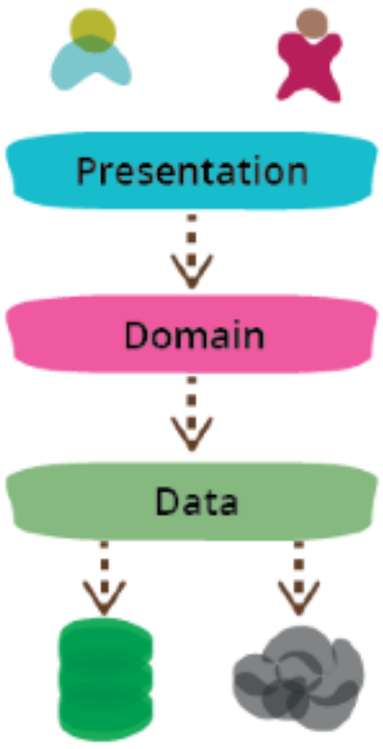
So what does the architecture of your application scream? When you look at the top level directory structure, and the source files in the highest level package; do they scream: **Health Care System**, or **Accounting System**, or **Inventory Management System**? Or do they scream: **Rails**, or **Spring/Hibernate**, or **ASP**?

The Theme of an Architecture

PresentationDomainDataLayering

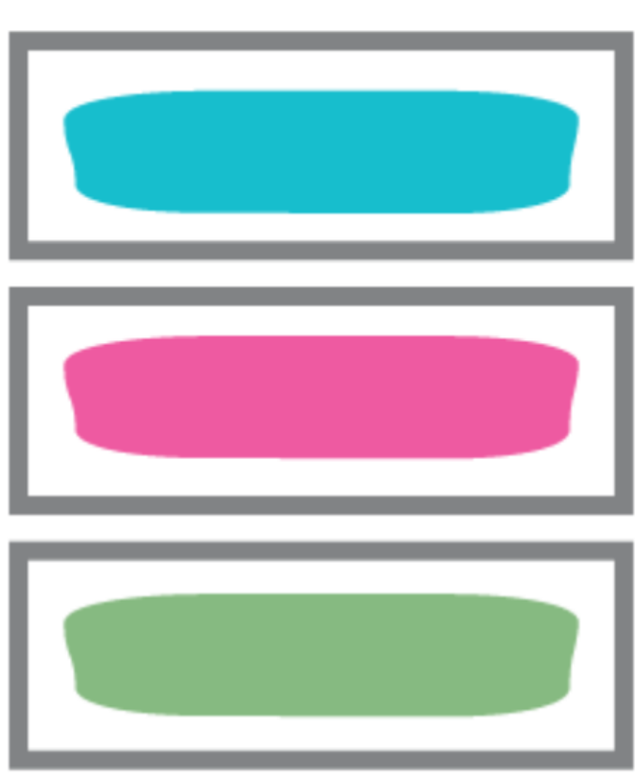
Martin Fowler
26 August 2015

One of the most common ways to modularize an information-rich to separate it into three broad layers: presentation (UI), domain l business logic), and data access. So you often see web applicati into a web layer that knows about handling http requests and ren HTML, a business logic layer that contains validations and calcul a data access layer that sorts out how to manage persistant data database or remote services.

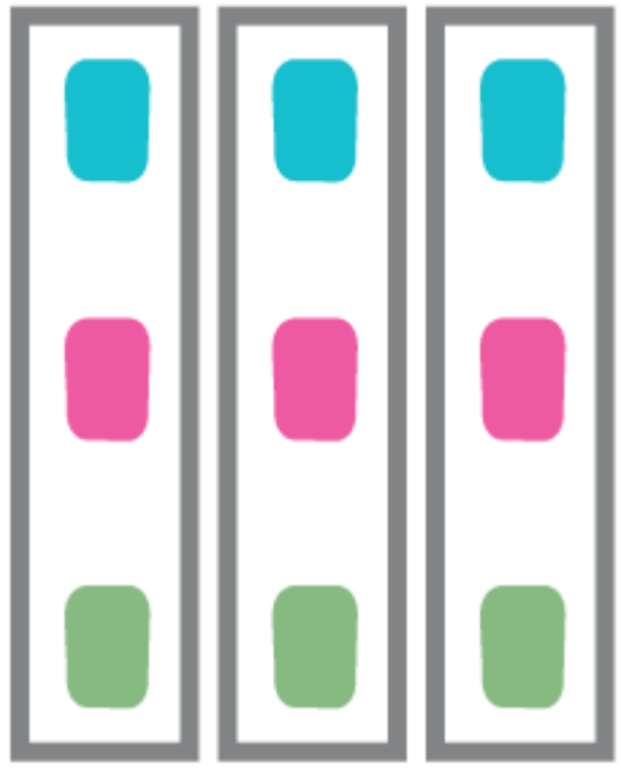


On the whole I've found this to be an effective form of modulariza many applications and one that I regularly use and encourage. It advantage (for me) is that it allows me to **reduce the scope of n**

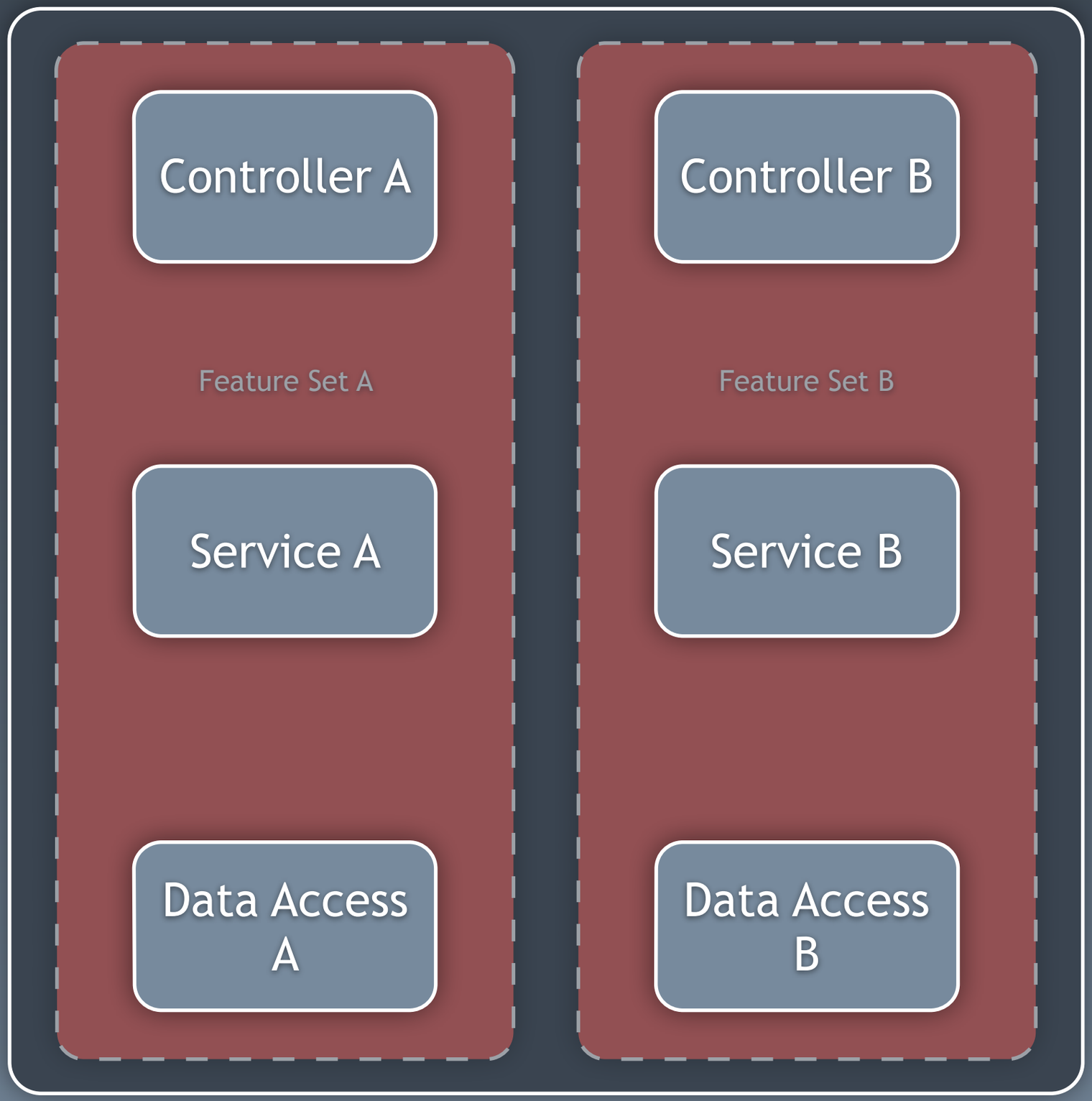
Although presentation-domain-data separation is a common approach, it should only be applied at a relatively small granularity. As an application grows, each layer can get sufficiently complex on its own that you need to modularize further. When this happens it's usually not best to use presentation-domain-data as the higher level of modules. Often frameworks encourage you to have something like view-model-data as the top level namespaces; that's ok for smaller systems, but once any of these layers gets too big you should split your top level into domain oriented modules which are internally layered.



Don't use layers as the top level modules in a complex application...



... instead make your top level modules be full-stack



Package by feature (vertical slicing)

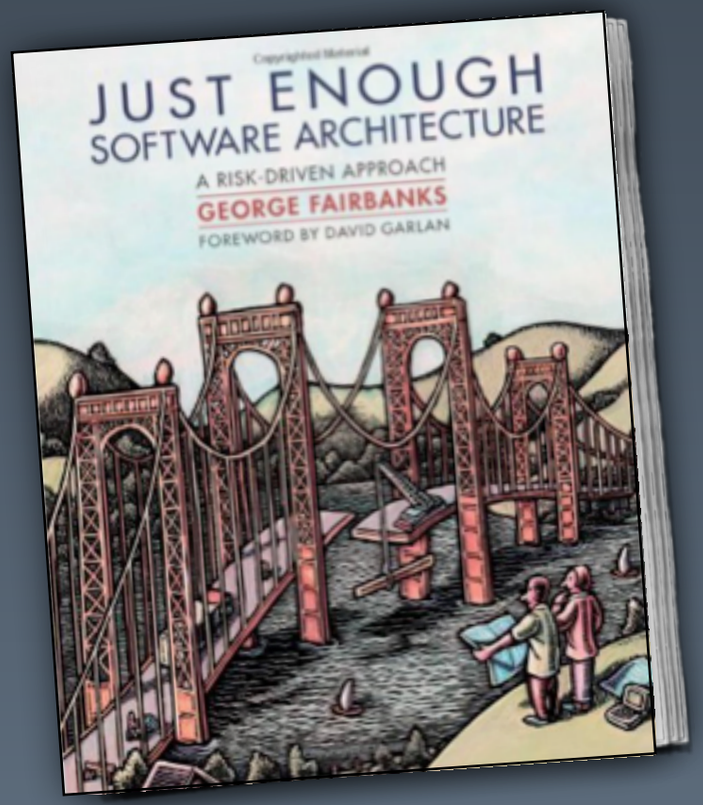
The intersection of
software architecture
and code

Abstractions

on diagrams

should reflect the

code



“architecturally-evident coding style”

Architecturally-evident coding styles include:

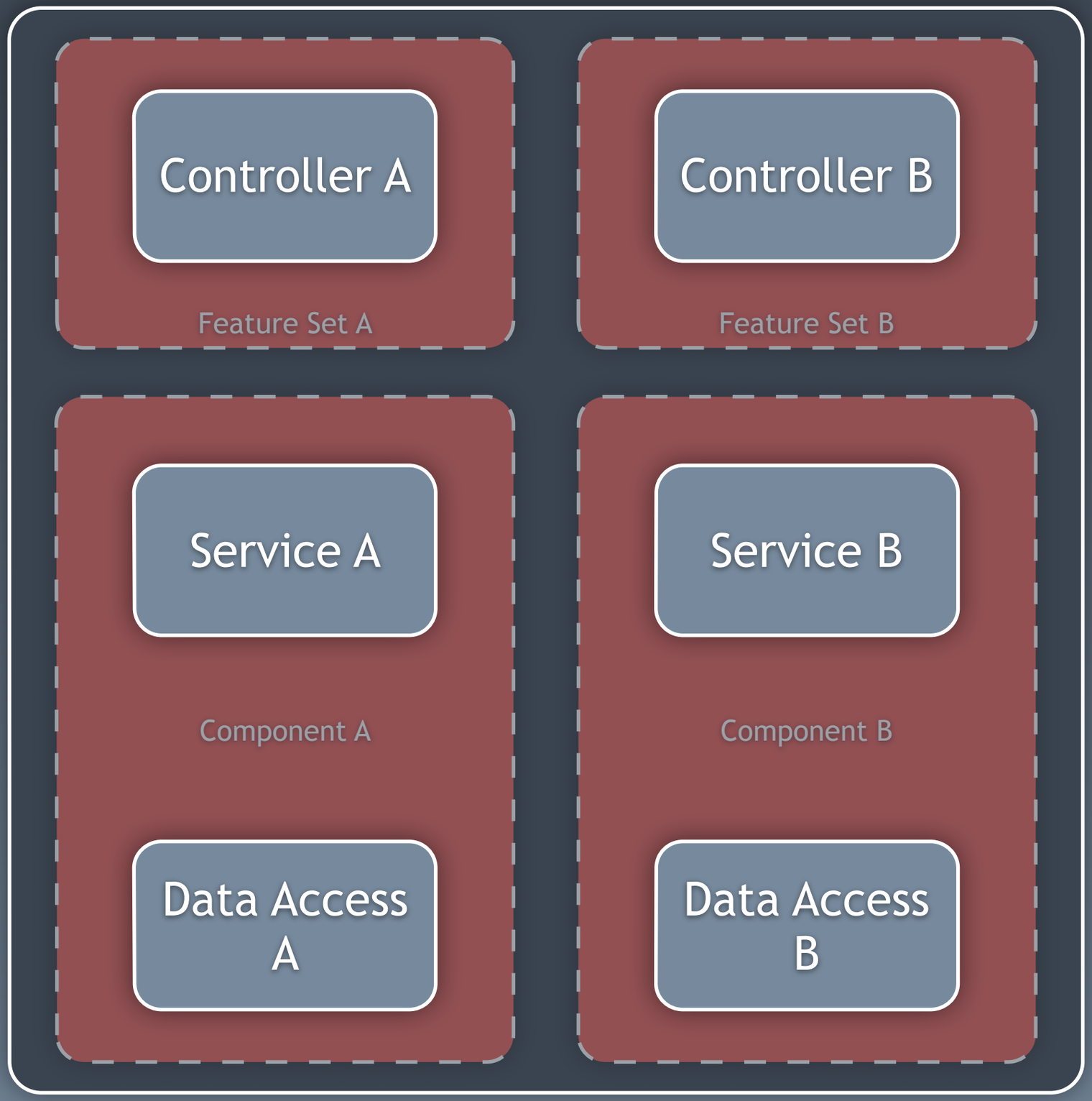
Annotations/attributes (`@Component`, `[Component]`, etc)

Naming conventions (`*Service`)

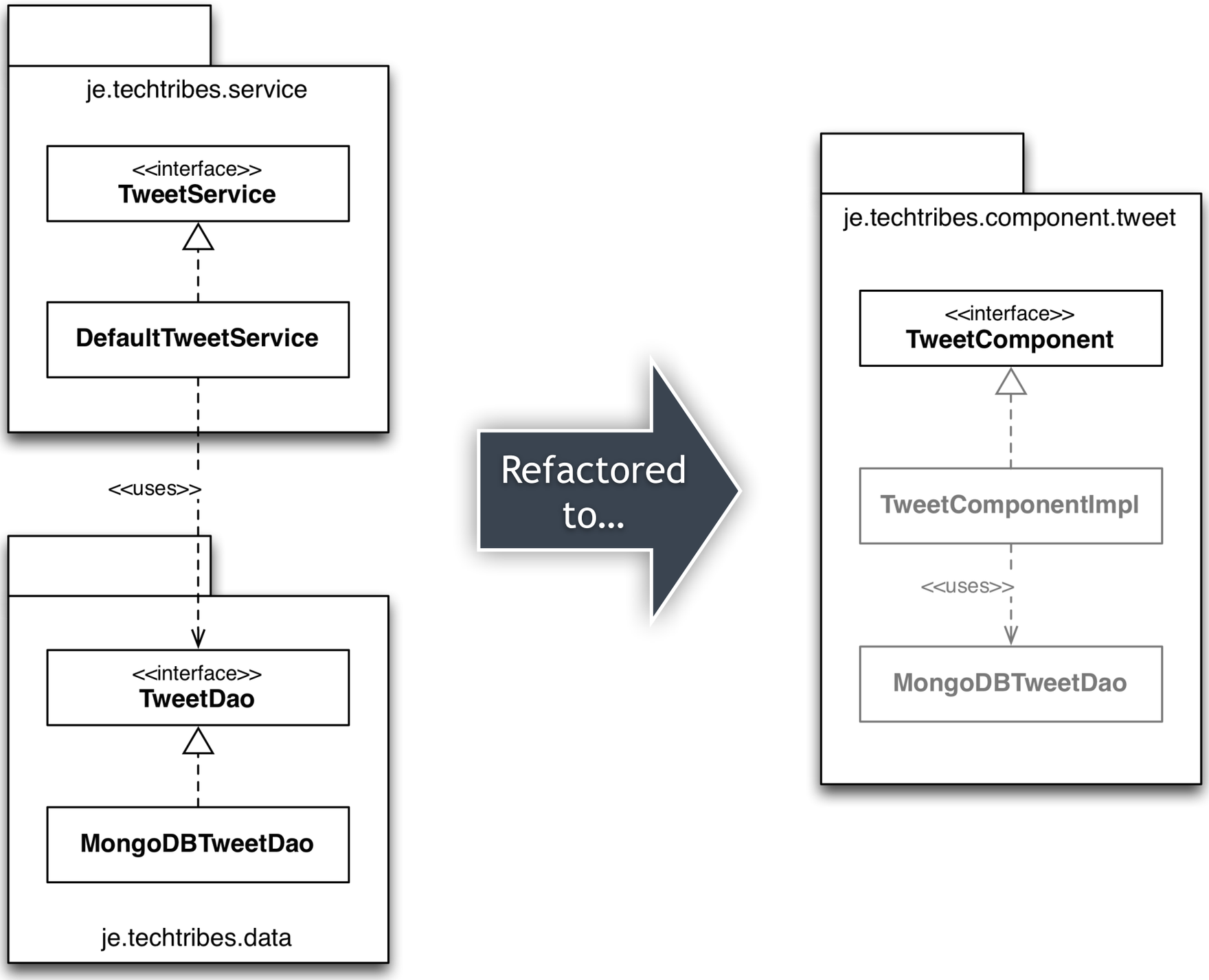
Namespacing/packaging

(`com.mycompany.system.components.*`)

Maven modules, OSGi modules, Java 9 and Jigsaw, JavaScript module patterns, ECMAScript 6 modules, microservices, etc



Package by component



An example of an architecturally-evident coding style

Project

techtribesje [techtribes] (~/sandbox/te
- docs
- lib
  - production
  - runtime
  - test
- structurizr
- techtribes-core
  - sql
  - src
    - je
      - techtribes
        - component
          - activity
          - badge
          - book
          - contentsource
          - creation
          - event
          - github
          - log
          - newsfeedentry
          - search
          - tweet
            - component.xml
            - MongoDbTweetDao
            - TweetComponent
            - TweetComponentImpl
            - TweetException
          - util
        - config.xml
  - test
  - techtribes-core.iml
- techtribes-updater
- techtribes-util
- techtribes-web
- .gitignore
- build.number

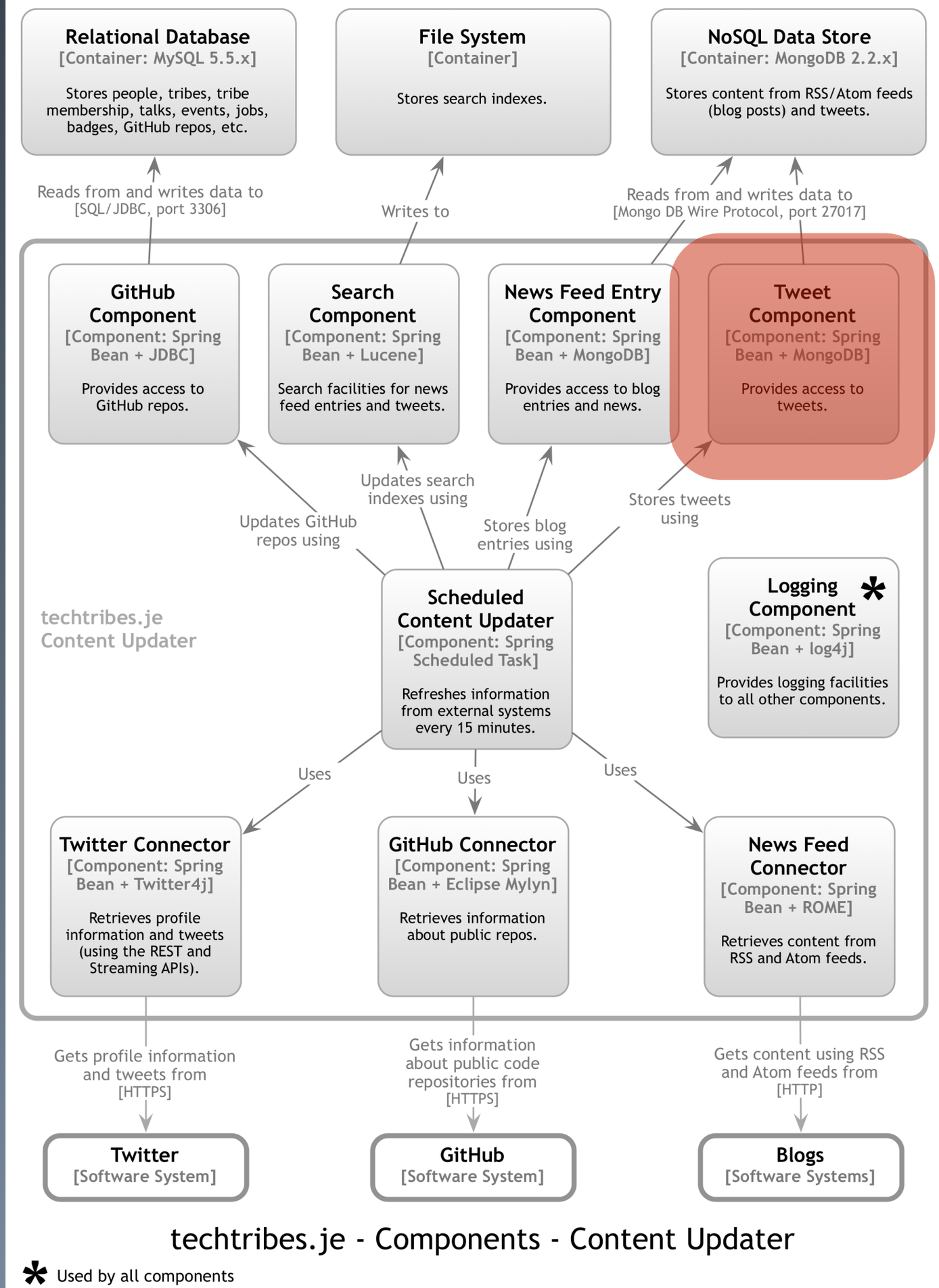
```
package je.techtribes.component.tweet;

import ...

/**
 * Provides access to tweets.
 */
@Component
public interface TweetComponent {

    /**
     * Gets the most recent tweets by page num
     */
    List<Tweet> getRecentTweets(int page, int

    /**
     * Gets the most recent tweets for a given
     */
    List<Tweet> getRecentTweets(ContentSource

    /**
     * Gets the most recent tweets for a given
     */
    List<Tweet> getRecentTweets(Collection<Cor

    /**
     * Calculates how many tweets a given pers
```

Context diagram
(level 1)

Container diagram
(level 2)

Component diagram
(level 3)

Class diagram
(level 4)



techtribes.je - Components - Content Updater

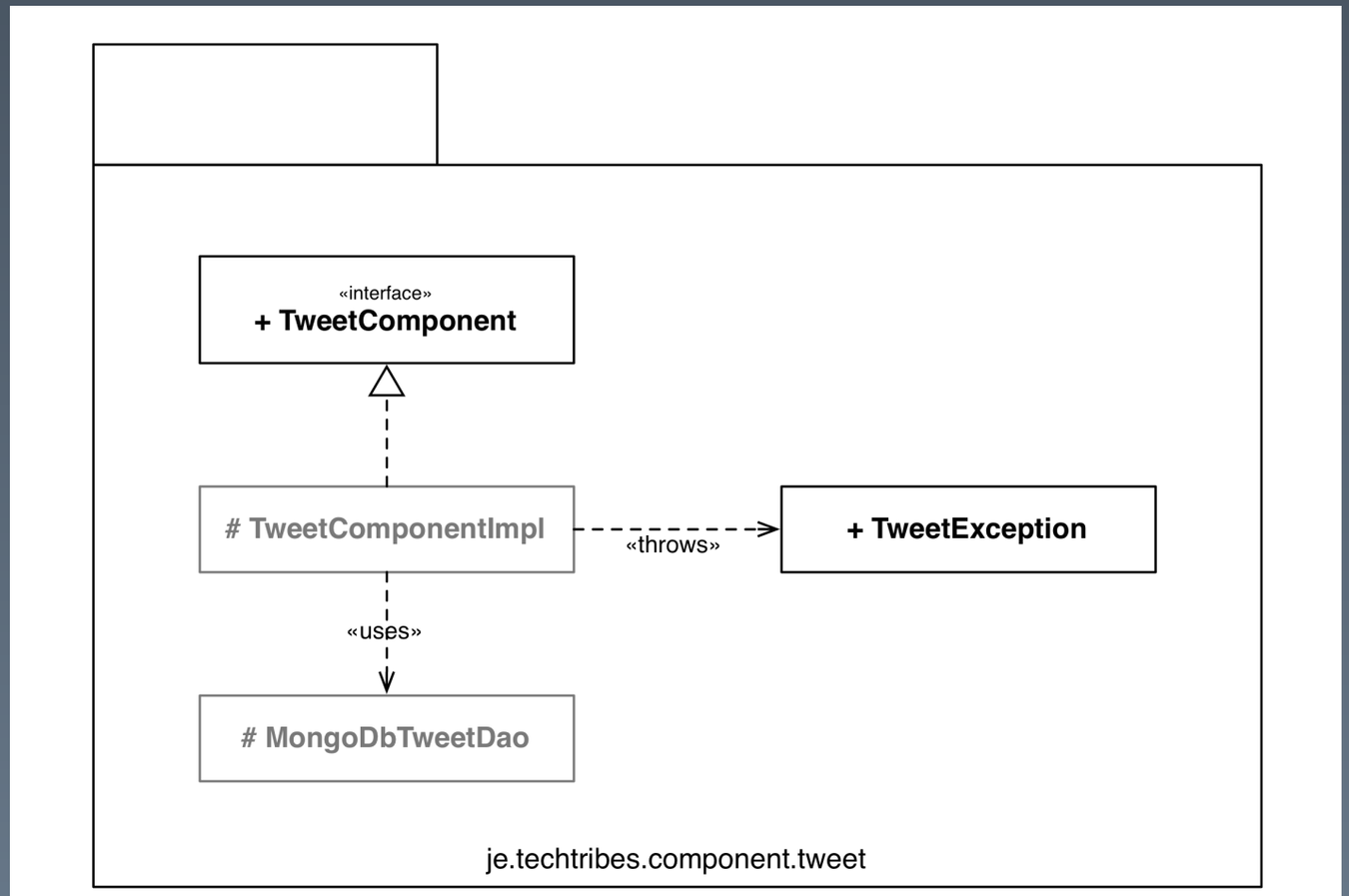
* Used by all components

Context diagram
(level 1)

Container diagram
(level 2)

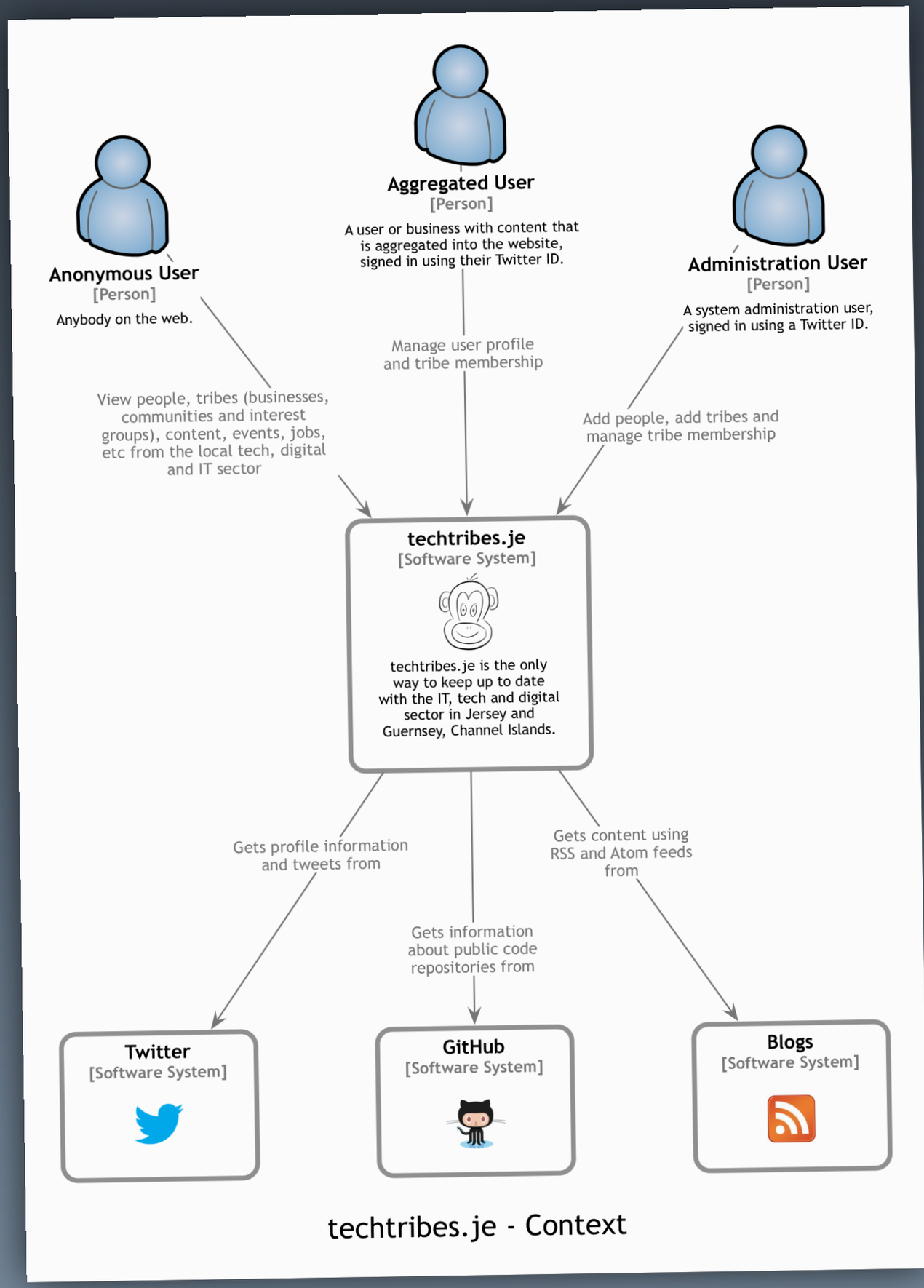
Component diagram
(level 3)

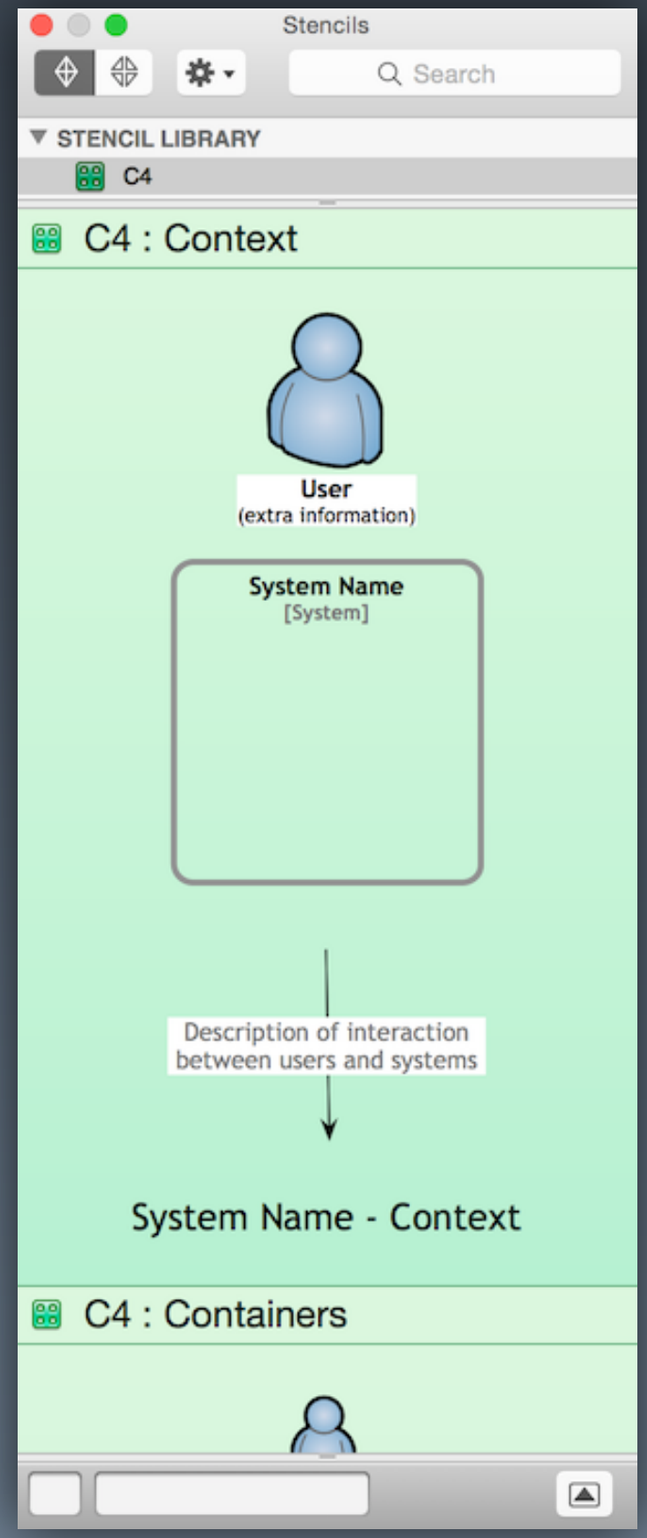
Class diagram
(level 4)



Modularity as a principle

Why?


Aggregated User
[Person]
A user or business with content that is aggregated into the website, signed in using their Twitter ID.
Anonymous User
[Person]
Anybody on the web.
Administration User
[Person]
A system administration user, signed in using a Twitter ID.
Manage user profile and tribe membership
View people, tribes (businesses, communities and interest groups), content, events, jobs, etc from the local tech, digital and IT sector
Add people, add tribes and manage tribe membership
techtribes.je
[Software System]
techtribes.je is the only way to keep up to date with the IT, tech and digital sector in Jersey and Guernsey, Channel Islands.
Gets profile information and tweets from
Gets information about public code repositories from
Gets content using RSS and Atom feeds from
Twitter
[Software System]
GitHub
[Software System]
Blogs
[Software System]
techtribes.je - Context


Stencils
Search
STENCIL LIBRARY
C4
C4 : Context
User
(extra information)
System Name
[System]
Description of interaction between users and systems
System Name - Context
C4 : Containers

Tools?

Sketches get out of date,
so why not
auto-generate
the diagrams?

Diagramming tools see

packages

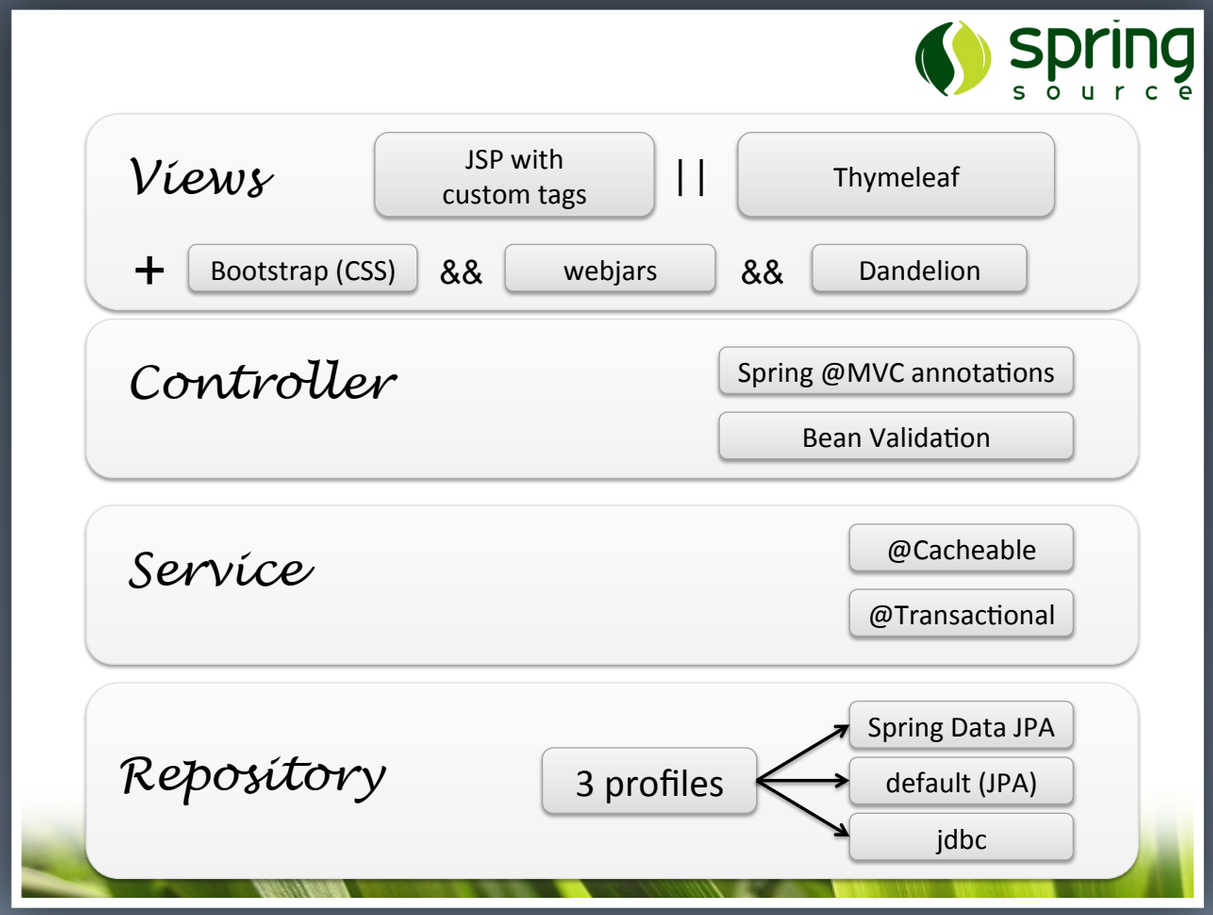
and classes

rather than components

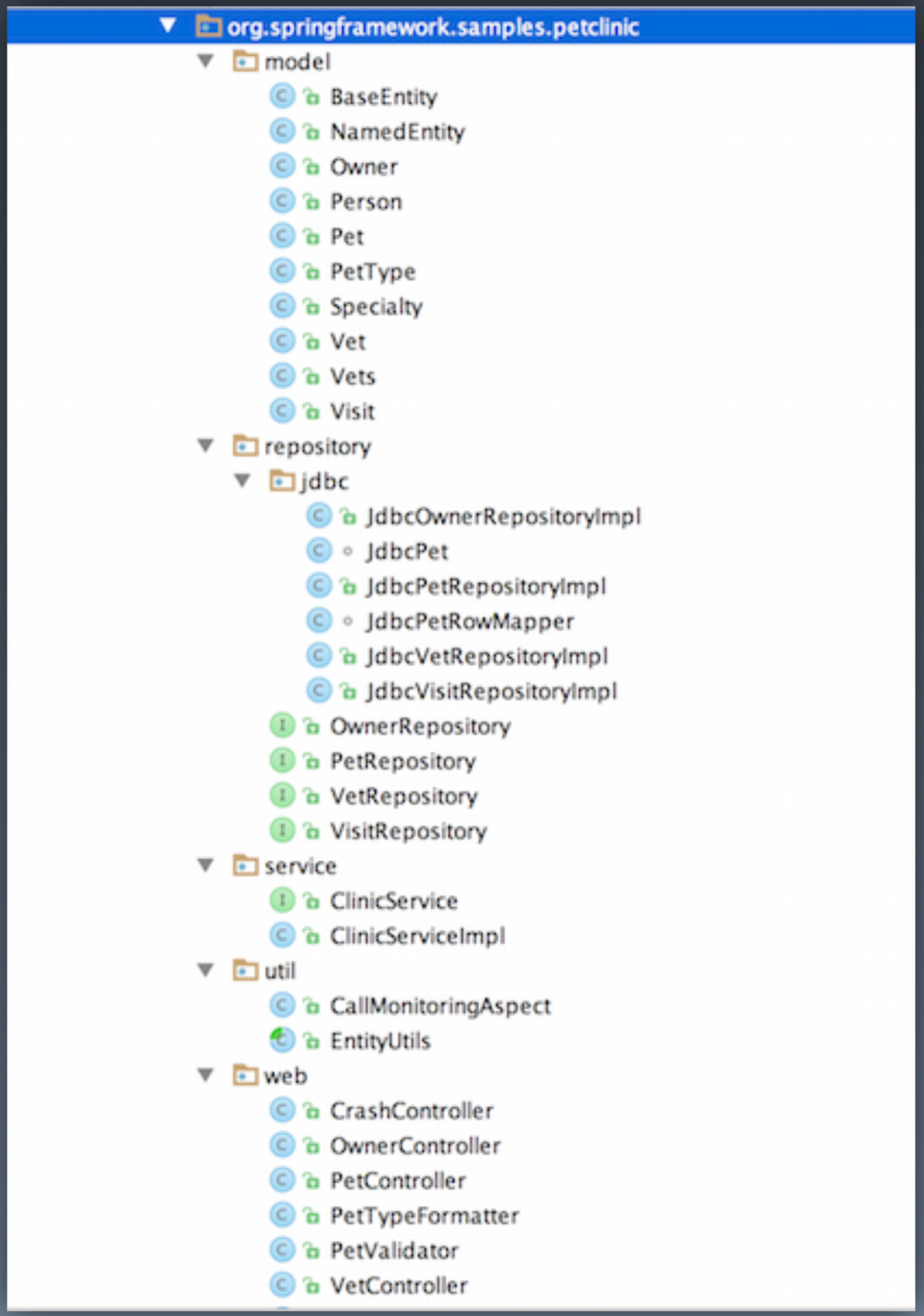
What is a “component”?

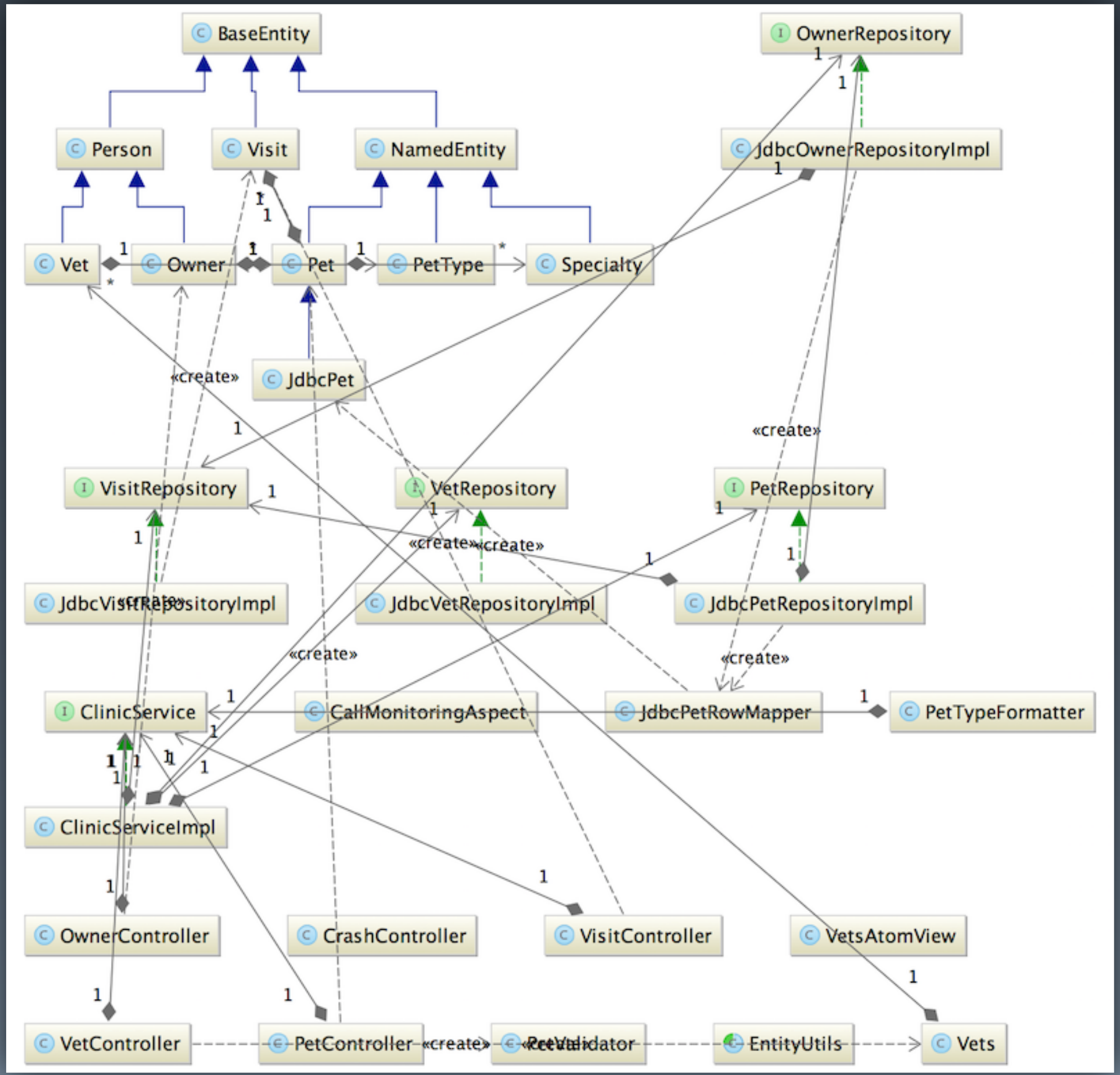
Spring PetClinic

https://github.com/spring-projects/spring-petclinic/



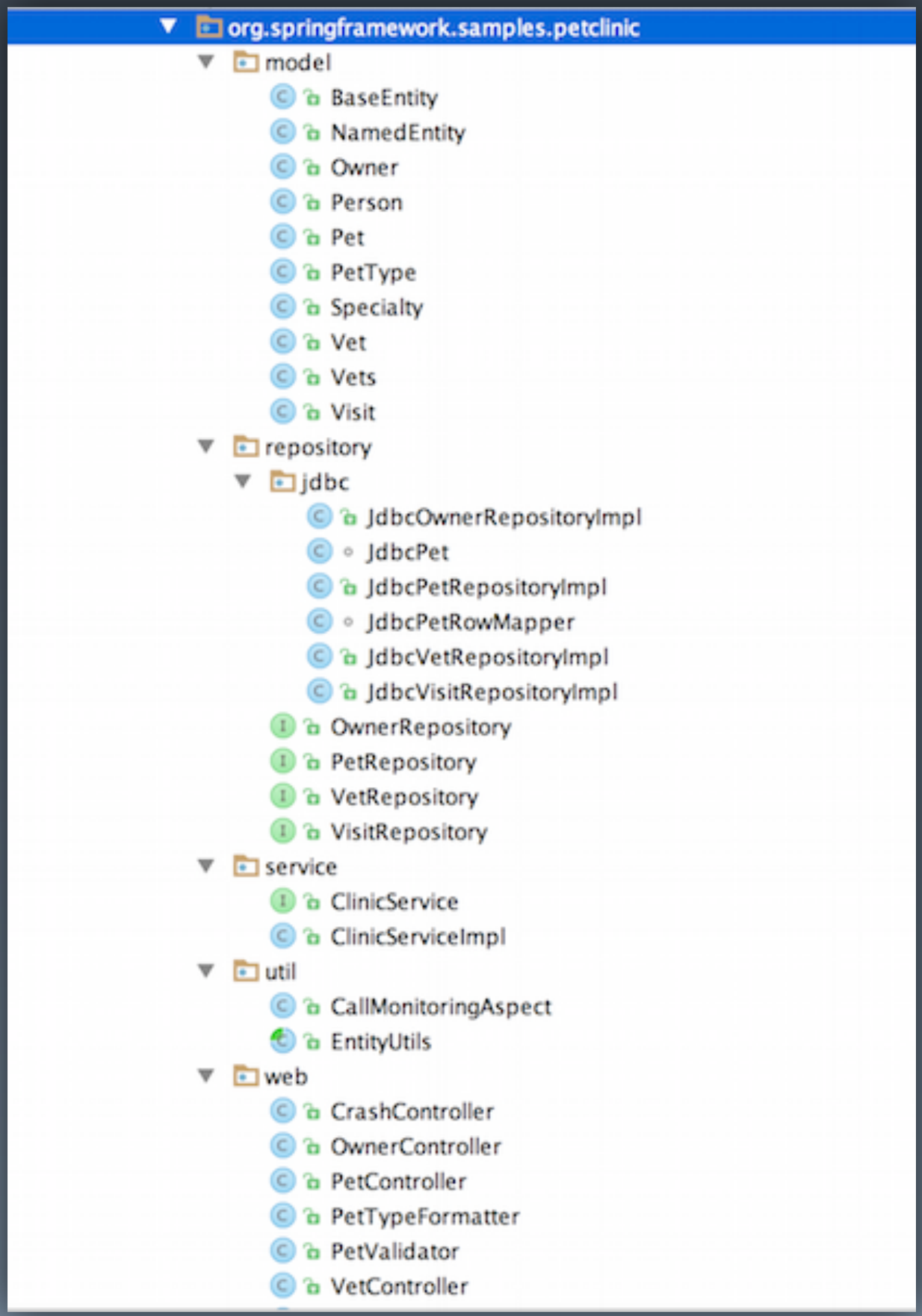
https://speakerdeck.com/michaelisvy/spring-petclinic-sample-application

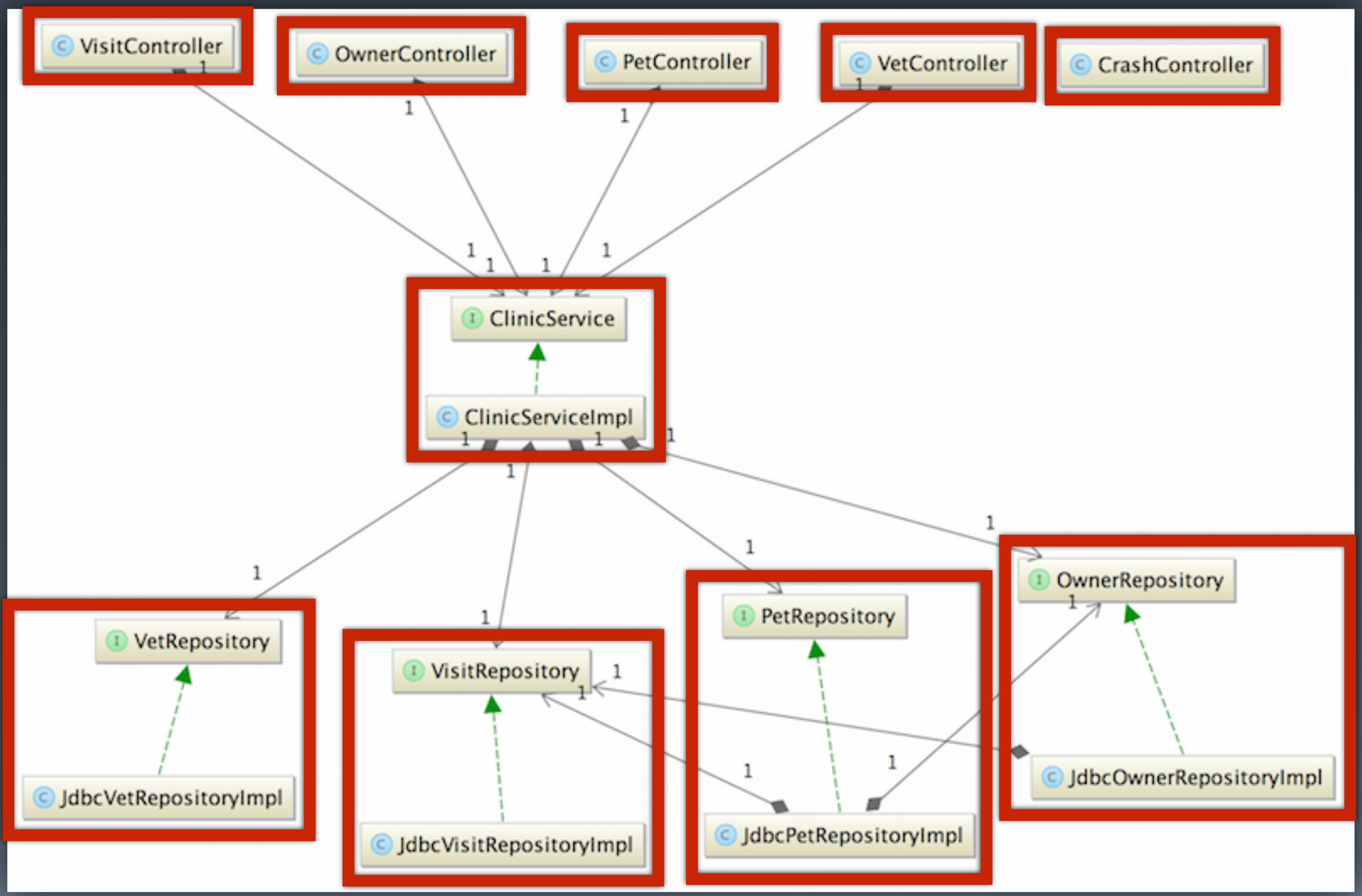




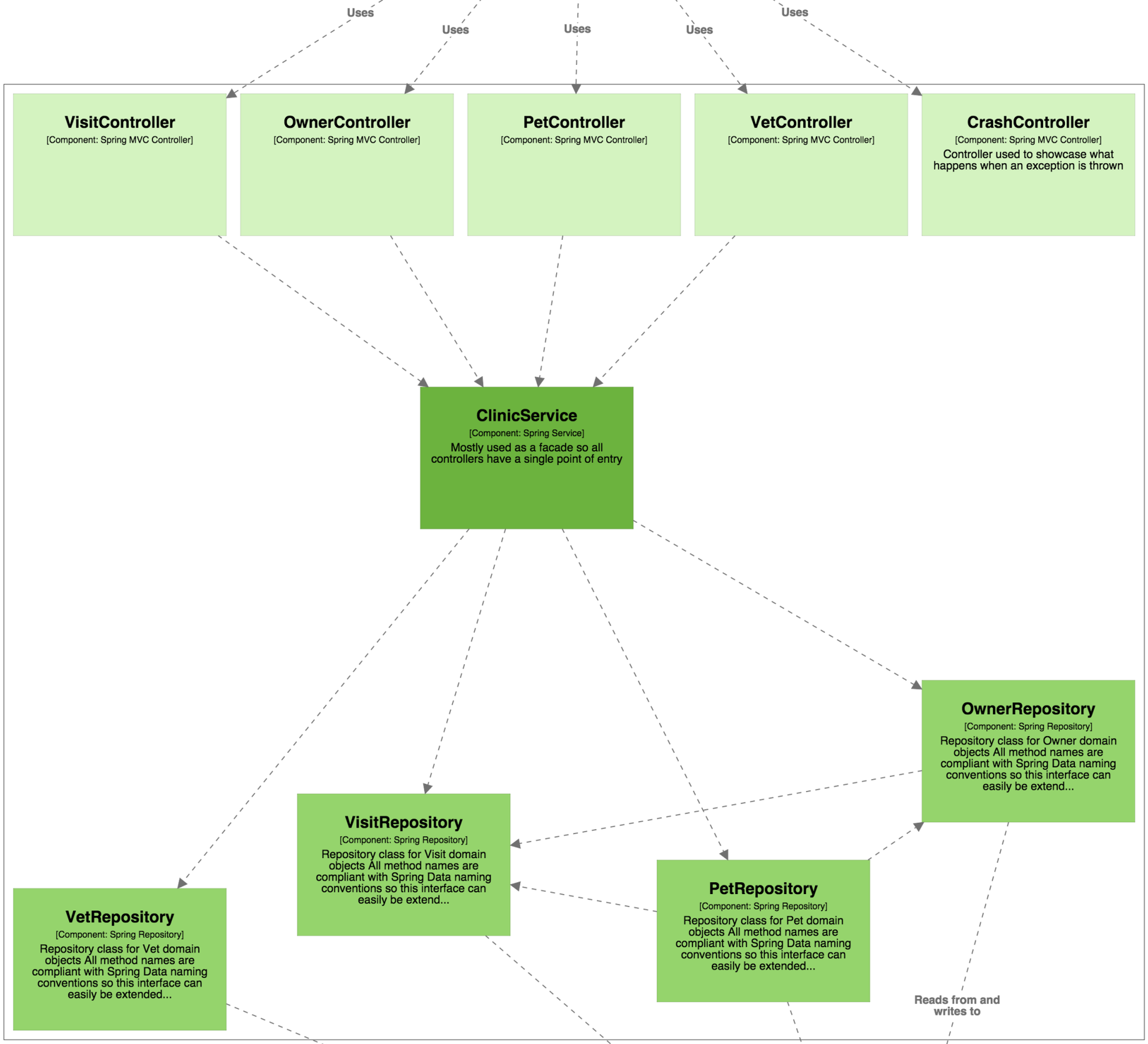
An auto-generated UML class diagram

What are the architecturally significant elements?





A UML class diagram showing architecturally significant elements



A component diagram, based upon the code

The code is the

embodiment

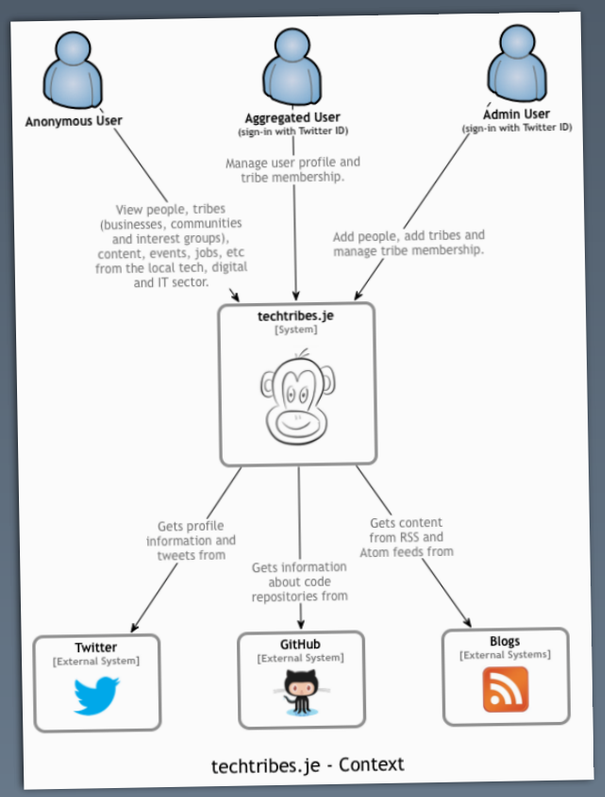
of the architecture

Is the architecture *in* the code?

In practice, architecture is embodied and ***recoverable from code****, and many languages provide architecture-level views of the system.*

A Survey of Architecture Description Languages
by Paul C. Clements

Context



People

Security groups/roles in configuration files, etc.

Software Systems

Integration points, APIs, known libraries, credentials for inbound consumers, etc.

Containers

IDE projects/modules, build output (code and infrastructure), etc.

Components

Extractable from the code if an architecturally-evident coding style has been adopted.

People

Security groups/roles in configuration files, etc.

Software Systems

Integration points, APIs, known libraries, credentials for inbound consumers, etc.

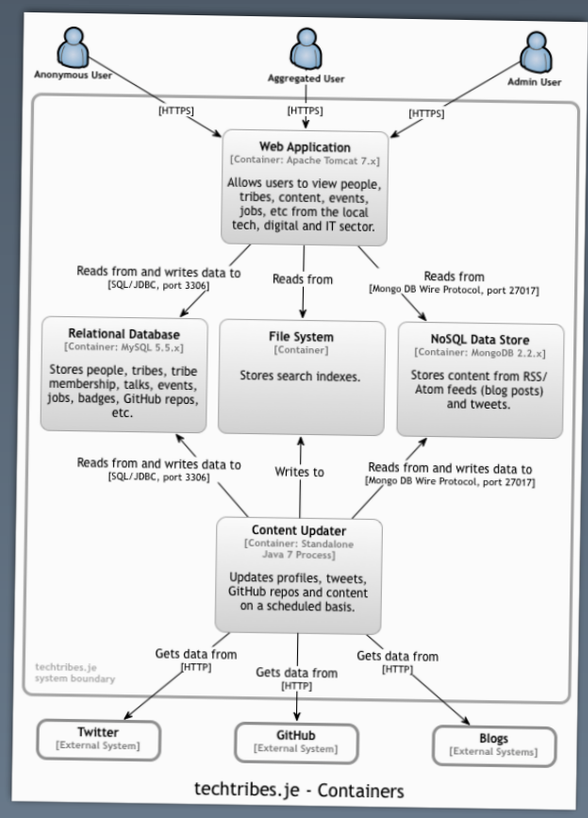
Containers

IDE projects/modules, build output (code and infrastructure), etc.

Components

Extractable from the code if an architecturally-evident coding style has been adopted.

Containers



People

Security groups/roles in configuration files, etc.

Software Systems

Integration points, APIs, known libraries, credentials for inbound consumers, etc.

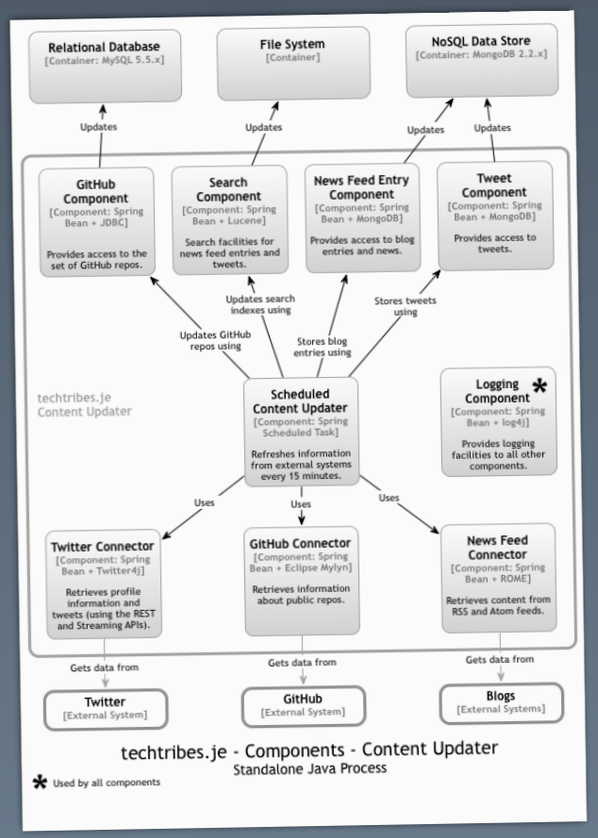
Containers

IDE projects/modules, build output (code and infrastructure), etc.

Components

Extractable from the code if an architecturally-evident coding style has been adopted.

Components



Extract as much of the software architecture from the code as possible, and supplement where necessary

Create an architecture description language using code

```java
/**
 * This is a C4 representation of the Spring PetClinic sample app
 * (https://github.com/spring-projects/spring-petclinic/).
 *
 * Use the examples/springpetclinic.sh file to run this example -
 * you'll need a compiled version of the app on the CLASSPATH.
 */
public class SpringPetClinic {

    public static void main(String[] args) throws Exception {
        Workspace workspace = new Workspace("Spring PetClinic",
                "This is a C4 representation of the Spring PetClinic sample app (https://github.com/spring-pro
        Model model = workspace.getModel();

        // create the basic model (the stuff we can't get from the code)
        SoftwareSystem springPetClinic = model.addSoftwareSystem("Spring PetClinic", "Allows employees to view
        Person user = model.addPerson("Clinic Employee", "An employee of the clinic");
        user.uses(springPetClinic, "Uses");

        Container webApplication = springPetClinic.addContainer(
                "Web Application", "Allows employees to view and manage information regarding the veterinarians
        Container relationalDatabase = springPetClinic.addContainer(
                "Relational Database", "Stores information regarding the veterinarians, the clients, and their
        user.uses(webApplication, "Uses", "HTTP");
        webApplication.uses(relationalDatabase, "Reads from and writes to", "JDBC, port 9001");

        // and now automatically find all Spring @Controller, @Component, @Service and @Repository components
        ComponentFinder componentFinder = new ComponentFinder(
                webApplication, "org.springframework.samples.petclinic",
                new SpringComponentFinderStrategy(),
                new JavadocComponentFinderStrategy(new File("/Users/simon/Documents/sandbox/spring/spring-petc
        componentFinder.findComponents();

        // connect the user to all of the Spring MVC controllers
        webApplication.getComponents().stream()
                .filter(c -> c.getTechnology().equals("Spring MVC Controller"))
                .forEach(c -> user.uses(c, "Uses"));

        // connect all of the repository components to the relational database
```

```
// and now automatically find all Spring @Controller, @Component, @Service and @Repository components
ComponentFinder componentFinder = new ComponentFinder(
        webApplication, "org.springframework.samples.petclinic",
        new SpringComponentFinderStrategy(),
        new JavadocComponentFinderStrategy(new File("/Users/simon/Documents/sandbox/spring/spring-petc
componentFinder.findComponents();

// connect the user to all of the Spring MVC controllers
webApplication.getComponents().stream()
        .filter(c -> c.getTechnology().equals("Spring MVC Controller"))
        .forEach(c -> user.uses(c, "Uses"));

// connect all of the repository components to the relational database
webApplication.getComponents().stream()
        .filter(c -> c.getTechnology().equals("Spring Repository"))
        .forEach(c -> c.uses(relationalDatabase, "Reads from and writes to"));

// finally create some views
ViewSet viewSet = workspace.getViews();
SystemContextView contextView = viewSet.createContextView(springPetClinic);
contextView.addAllSoftwareSystems();
contextView.addAllPeople();

ContainerView containerView = viewSet.createContainerView(springPetClinic);
containerView.addAllPeople();
containerView.addAllSoftwareSystems();
containerView.addAllContainers();

ComponentView componentView = viewSet.createComponentView(webApplication);
componentView.addAllComponents();
componentView.addAllPeople();
componentView.add(relationalDatabase);

// link the architecture model with the code
for (Component component : webApplication.getComponents()) {
    if (component.getSourcePath() != null) {
        component.setSourcePath(component.getSourcePath().replace(
                "/Users/simon/Documents/sandbox/spring/spring-petclinic/",
```

```
// finally create some views
ViewSet viewSet = workspace.getViews();
SystemContextView contextView = viewSet.createContextView(springPetClinic);
contextView.addAllSoftwareSystems();
contextView.addAllPeople();

ContainerView containerView = viewSet.createContainerView(springPetClinic);
containerView.addAllPeople();
containerView.addAllSoftwareSystems();
containerView.addAllContainers();

ComponentView componentView = viewSet.createComponentView(webApplication);
componentView.addAllComponents();
componentView.addAllPeople();
componentView.add(relationalDatabase);
```

Structurizr for Java

(open source on GitHub)

GitHub This repository Search Explore Features Enterprise Blog Sign up Sign in

structurizr / java Watch 24 Star 59 Fork 30

Java tools

131 commits 1 branch 0 releases 3 contributors

branch: master java / +

Added a software architecture model for the Java EE Hands on Lab "Mov…

simonbrowndotje authored 14 days ago latest commit 90dea447f9

gradle	Gradle Support: cleanup workspace, commit gradle wrapper, remove ant/ivy	2 months ago
structurizr-annotations/src/co…	Some refactoring and an initial version of a Javadoc component finder…	25 days ago
structurizr-client	This saves a little bandwidth. :-)	14 days ago
structurizr-core	Fixed a bug where layout information wasn't getting copied when inter…	14 days ago
structurizr-examples	Added a software architecture model for the Java EE Hands on Lab "Mov…	14 days ago
structurizr-spring	Added a sourcePath property to components ... which is automatically …	23 days ago
.gitignore	Added some comments.	2 months ago
LICENSE	Initial commit	10 months ago
README.md	Added build instructions.	2 months ago
build.gradle	Changed styles -> configuration.styles	27 days ago
copyJars.sh	Added fixed some bugs and added an initial implementation to copy lay…	2 months ago
gradle.properties	Renamed components and messing with Maven repo publication.	2 months ago
gradlew	Trimmed down the Gradle config and stopped tracking the IDEA project …	2 months ago
gradlew.bat	Gradle support	2 months ago
settings.gradle	Renamed components and messing with Maven repo publication.	2 months ago

Code Issues 1 Pull requests 1 Pulse Graphs

HTTPS clone URL https://github.com

You can clone with HTTPS or Subversion.

Clone in Desktop Download ZIP

README.md

Structurizr for Java

Structurizr is an implementation of the C4 model as described in Simon Brown's Software Architecture for Developers book, which provides a way to easily and effectively communicate the software architecture of a software system. Structurizr allows you to create **software architecture models and diagrams as code**. This project contains the Java implementation and tooling.

Everything you see here is a work in progress. See www.structurizr.com for more information.

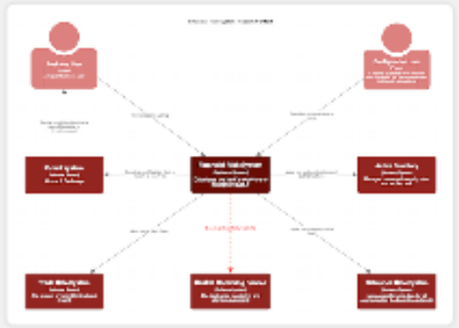
Building

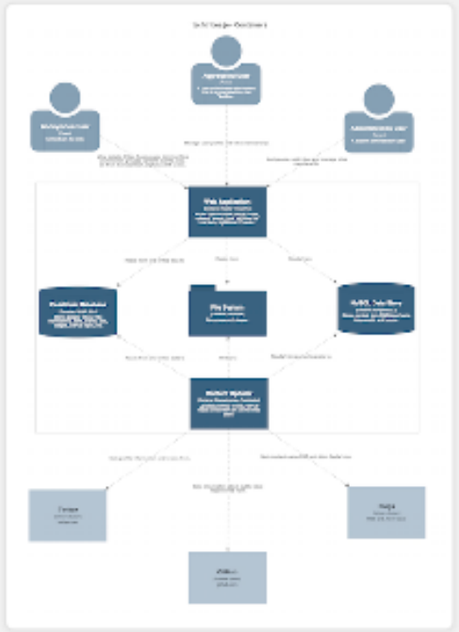
To build Struturizr for Java from the sources (you'll need Java 8)...

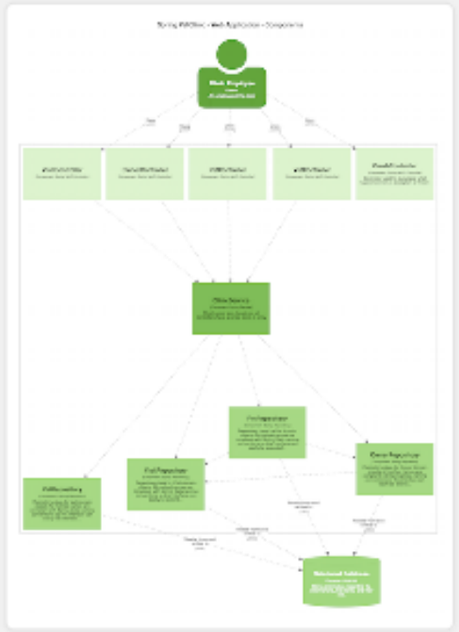
```
git clone https://github.com/structurizr/java.git

./gradlew build
```

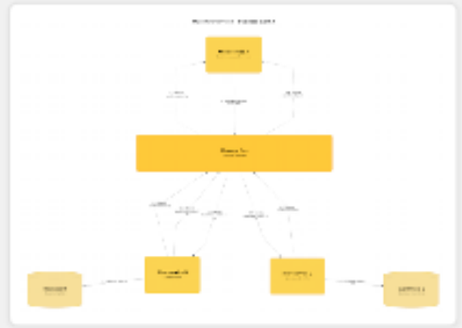
Structurizr

Create web-based software architecture diagrams using code.









Simple

No more endless hours spent messing with drawing tools, manually drawing boxes and lines. You create a software architecture model using code, Structurizr visualises it for you.

Versionable

The software architecture model is created using code, so it can be easily versioned alongside your production code.

Up-to-date

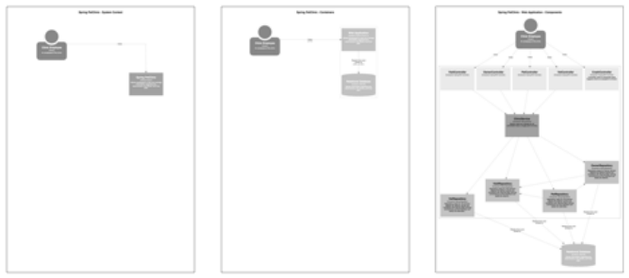
Integration with your build process means your software architecture model can be continuously kept up to date, especially if you extract parts of the model from your codebase.

Scalable

Static diagrams are hard to change and work with once they start to get large. Having the software architecture model as code puts you in control.

Structurizr Client

2. Upload the software architecture model to Structurizr using the web API



1. Write one or more programs to create a software architecture model of your software system

Manually create model elements using code or extract them from your software system by parsing source code or using reflection and static analysis techniques

Software System

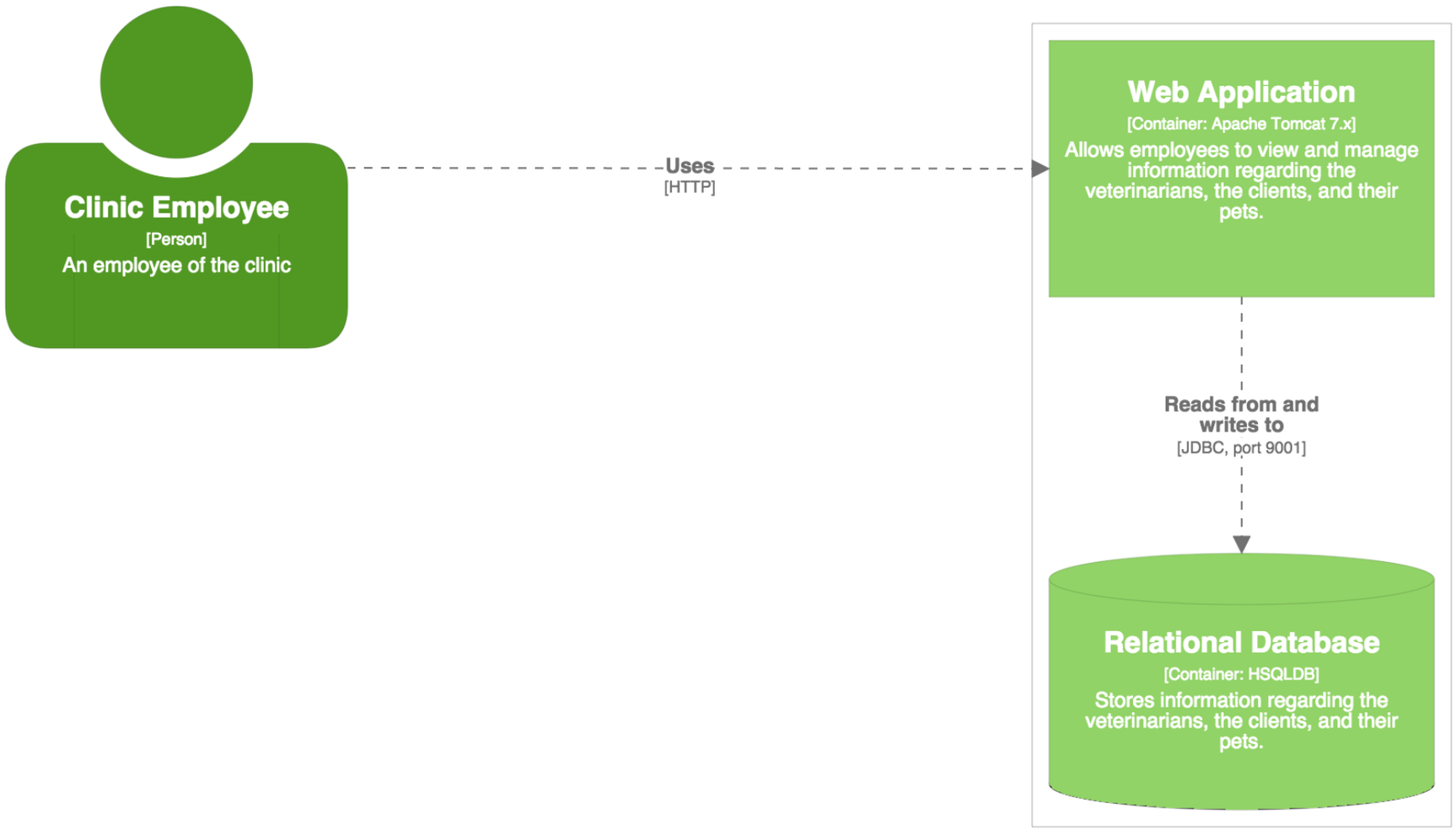
3. Explore, visualise and share your software architecture model with web-based software architecture diagrams

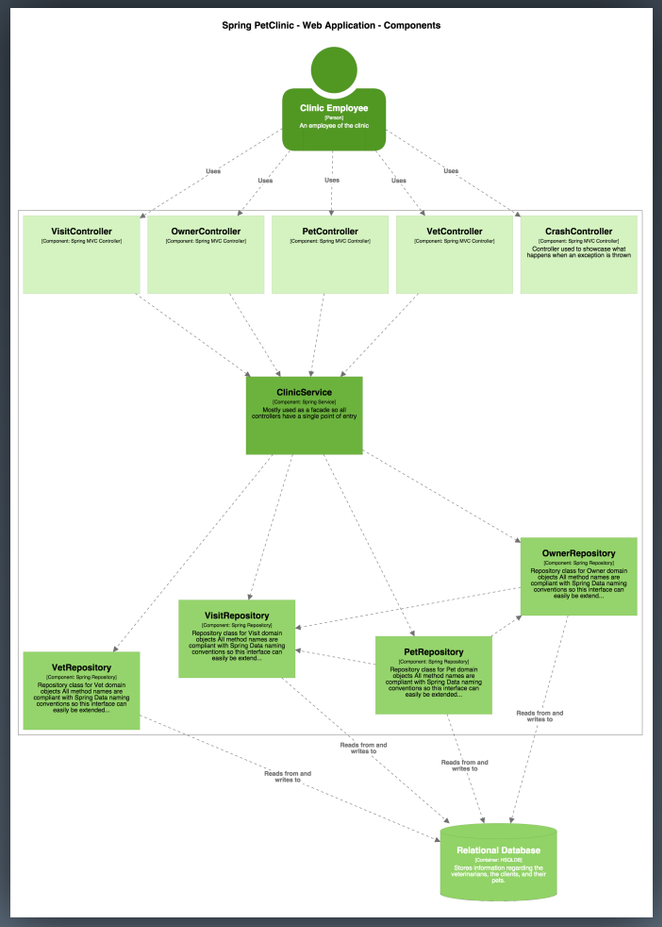
Stucturizr is a web-based tool that creates diagrams for you, based upon the software architecture model and views defined in your code

What is Structurizr?

Spring PetClinic - System Context

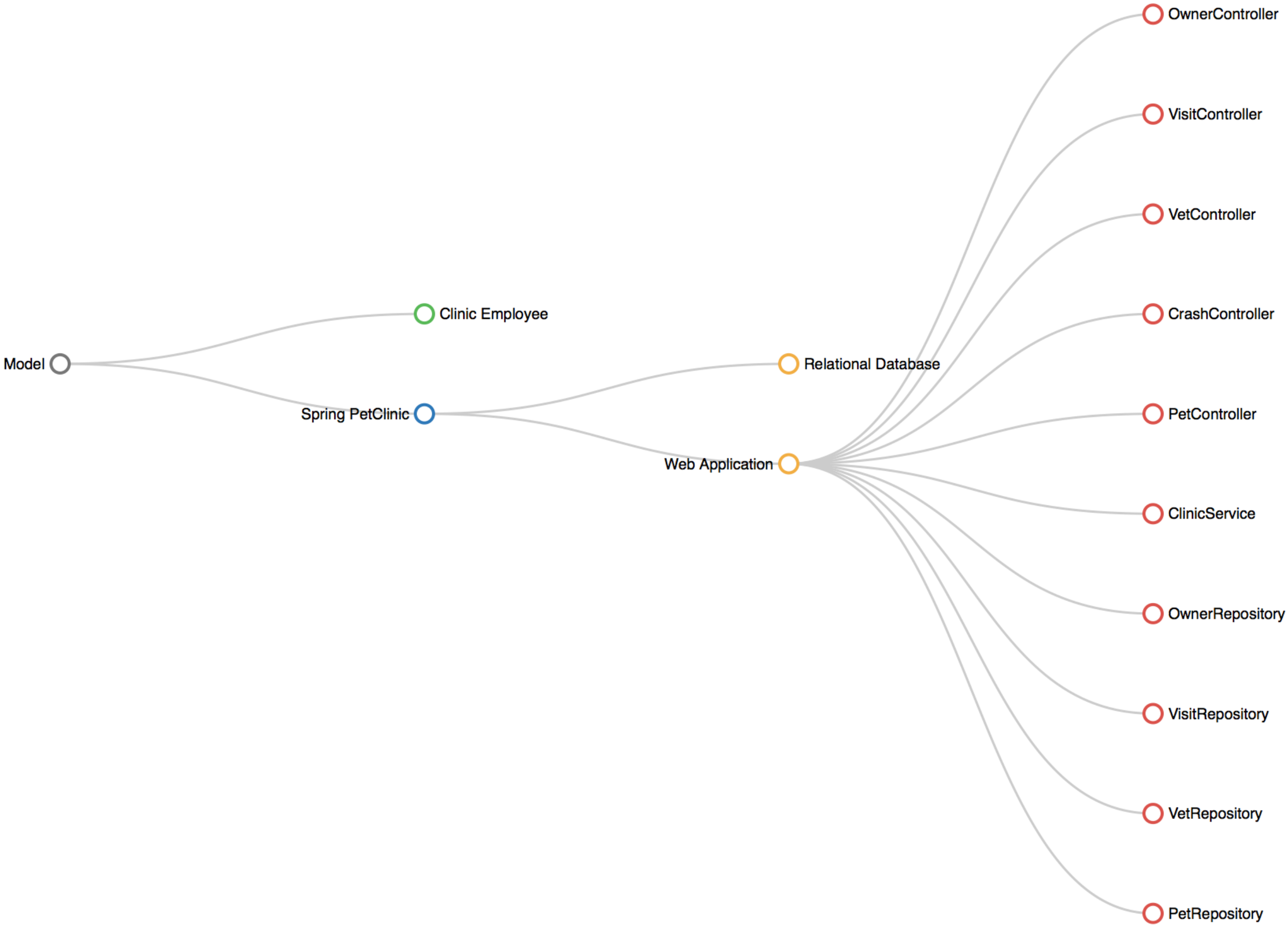
Spring PetClinic - Containers

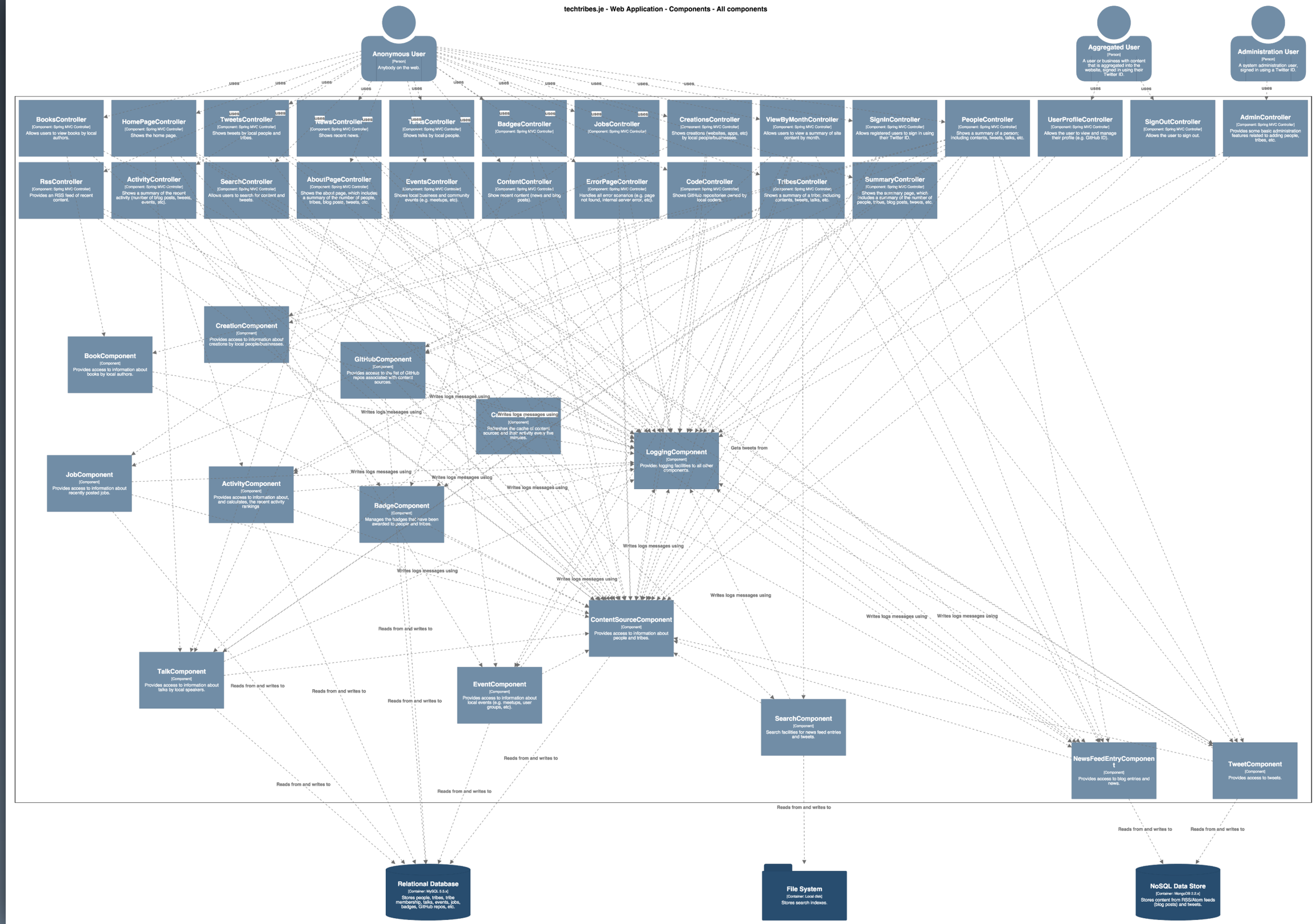




Spring PetClinic - Web Application - Components

Clinic Employee
[Person]
An employee of the clinic
Uses
VisitController
[Component: Spring MVC Controller]
OwnerController
[Component: Spring MVC Controller]
PetController
[Component: Spring MVC Controller]
VetController
[Component: Spring MVC Controller]
CrashController
[Component: Spring MVC Controller]
Controller used to showcase what happens when an exception is thrown
ClinicService
[Component: Spring Service]
Mostly used as a facade so all controllers have a single point of entry
OwnerRepository
[Component: Spring Repository]
Repository class for Owner domain objects All method names are compliant with Spring Data naming conventions so this interface can easily be extend...
VisitRepository
[Component: Spring Repository]
Repository class for Visit domain objects All method names are compliant with Spring Data naming conventions so this interface can easily be extend...
PetRepository
[Component: Spring Repository]
Repository class for Pet domain objects All method names are compliant with Spring Data naming conventions so this interface can easily be extended...
VetRepository
[Component: Spring Repository]
Repository class for Vet domain objects All method names are compliant with Spring Data naming conventions so this interface can easily be extended...
Reads from and writes to

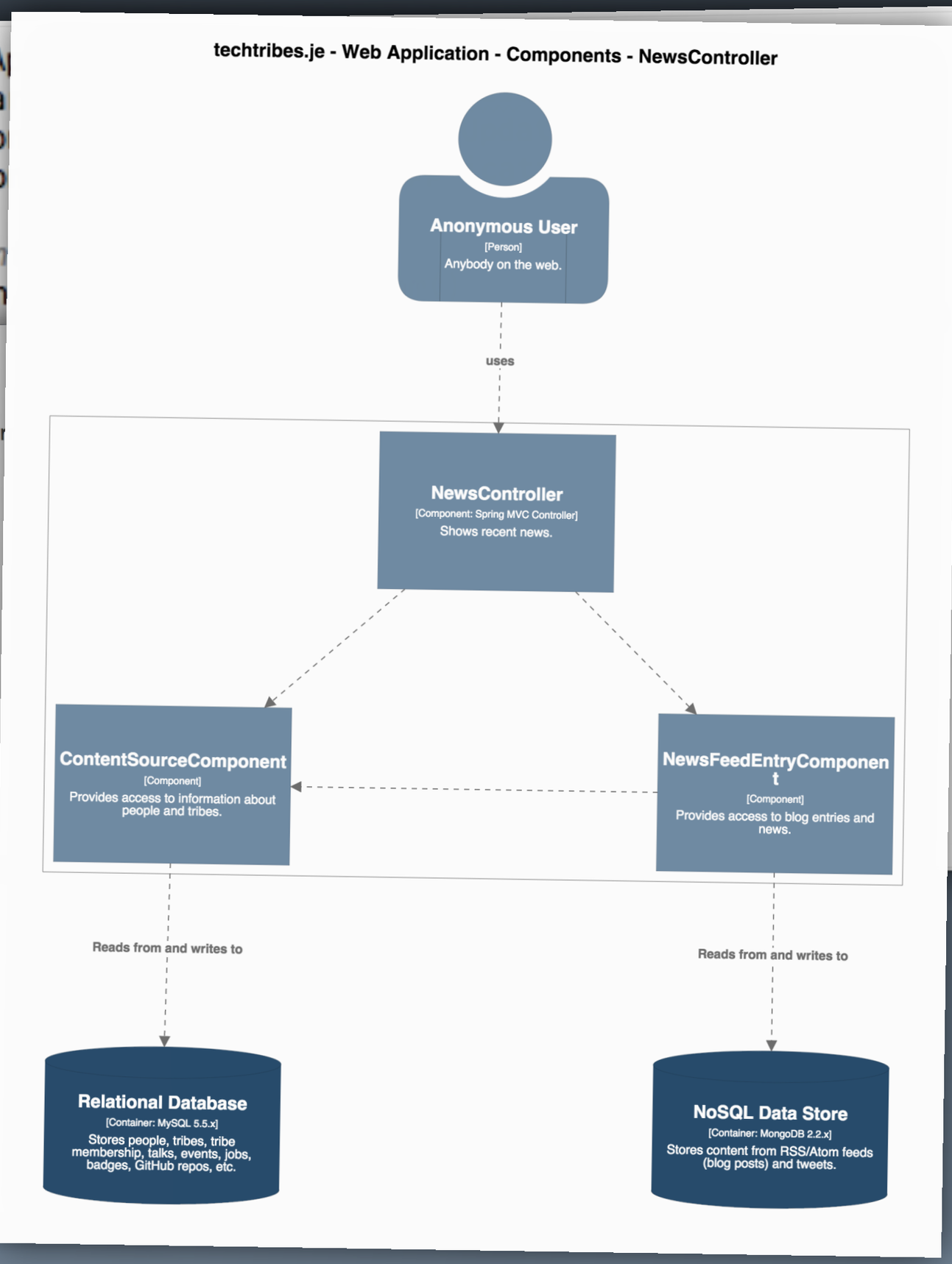

Model
Clinic Employee
Spring PetClinic
Relational Database
Web Application
OwnerController
VisitController
VetController
CrashController
PetController
ClinicService
OwnerRepository
VisitRepository
VetRepository
PetRepository



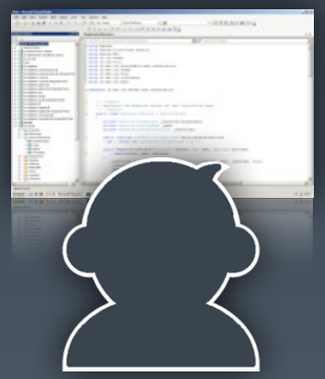
Does this approach scale?

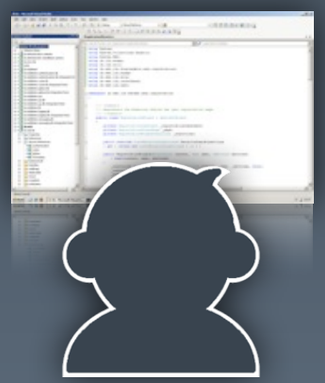
```
private static void createComponentViewsForWebA
    SoftwareSystem techTribes = model.getSoftwa
    Container contentUpdater = techTribes.getCo
    Container webApplication = techTribes.getCo

    // create one component view per Spring con
    Set<Component> controllers = webApplication
```

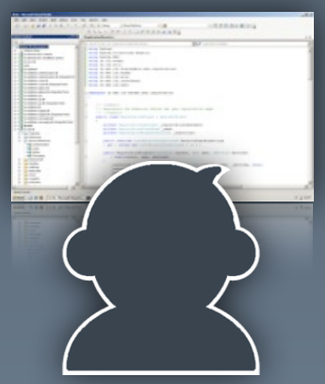
techtribes.je - System Context
techtribes.je - Containers
techtribes.je - Content Updater - Components - Awarding badges
techtribes.je - Content Updater - Components - Updating information from exter
techtribes.je - Content Updater - Components - Updating recent activity
techtribes.je - Web Application - Components - AboutPageController
techtribes.je - Web Application - Components - ActivityController
techtribes.je - Web Application - Components - AdminController
✓ techtribes.je - Web Application - Components - All components
techtribes.je - Web Application - Components - BadgesController
techtribes.je - Web Application - Components - BooksController
techtribes.je - Web Application - Components - CodeController
techtribes.je - Web Application - Components - ContentController
techtribes.je - Web Application - Components - CreationsController
techtribes.je - Web Application - Components - ErrorPageController
techtribes.je - Web Application - Components - EventsController
techtribes.je - Web Application - Components - HomePageController
techtribes.je - Web Application - Components - JobsController
techtribes.je - Web Application - Components - NewsController
techtribes.je - Web Application - Components - PeopleController
techtribes.je - Web Application - Components - RssController
techtribes.je - Web Application - Components - SearchController
techtribes.je - Web Application - Components - SignInController
techtribes.je - Web Application - Components - SignOutController
techtribes.je - Web Application - Components - SummaryController
techtribes.je - Web Application - Components - TalksController
techtribes.je - Web Application - Components - TribesController
techtribes.je - Web Application - Components - TweetsController
techtribes.je - Web Application - Components - UserProfileController
techtribes.je - Web Application - Components - ViewByMonthController



A model as code provides opportunities…







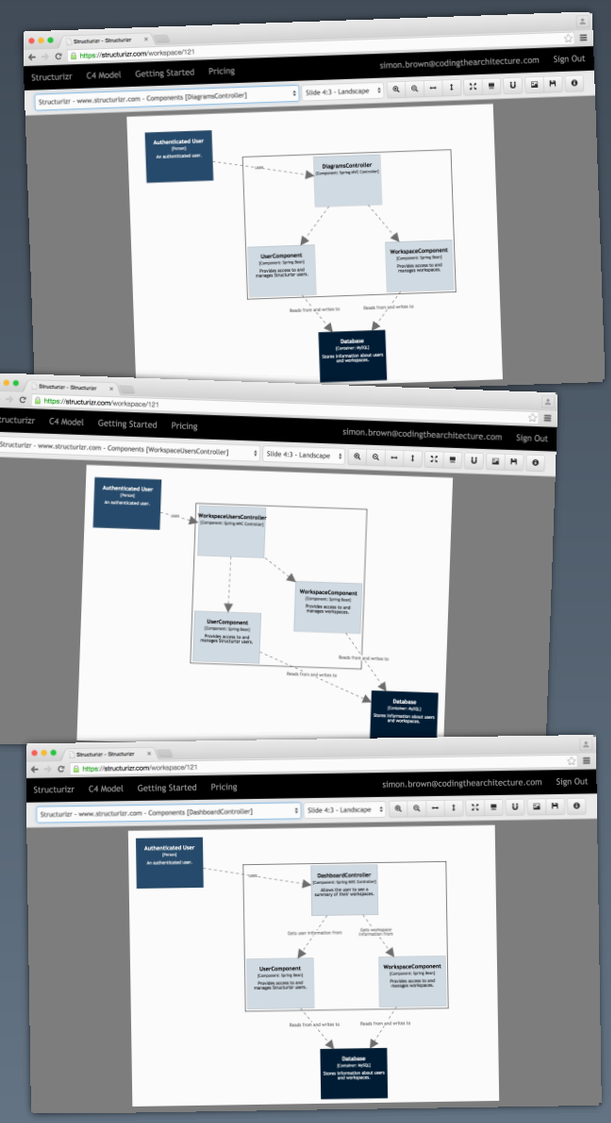
TeamCity

<APACHE ANT>

gradle

maven

Jenkins



…and build pipeline integration keeps software architecture models up-to-date

If the software architecture model is *in* the code, it can be extracted *from* the code

simon.brown@codingthearchitecture.com
@simonbrown on Twitter