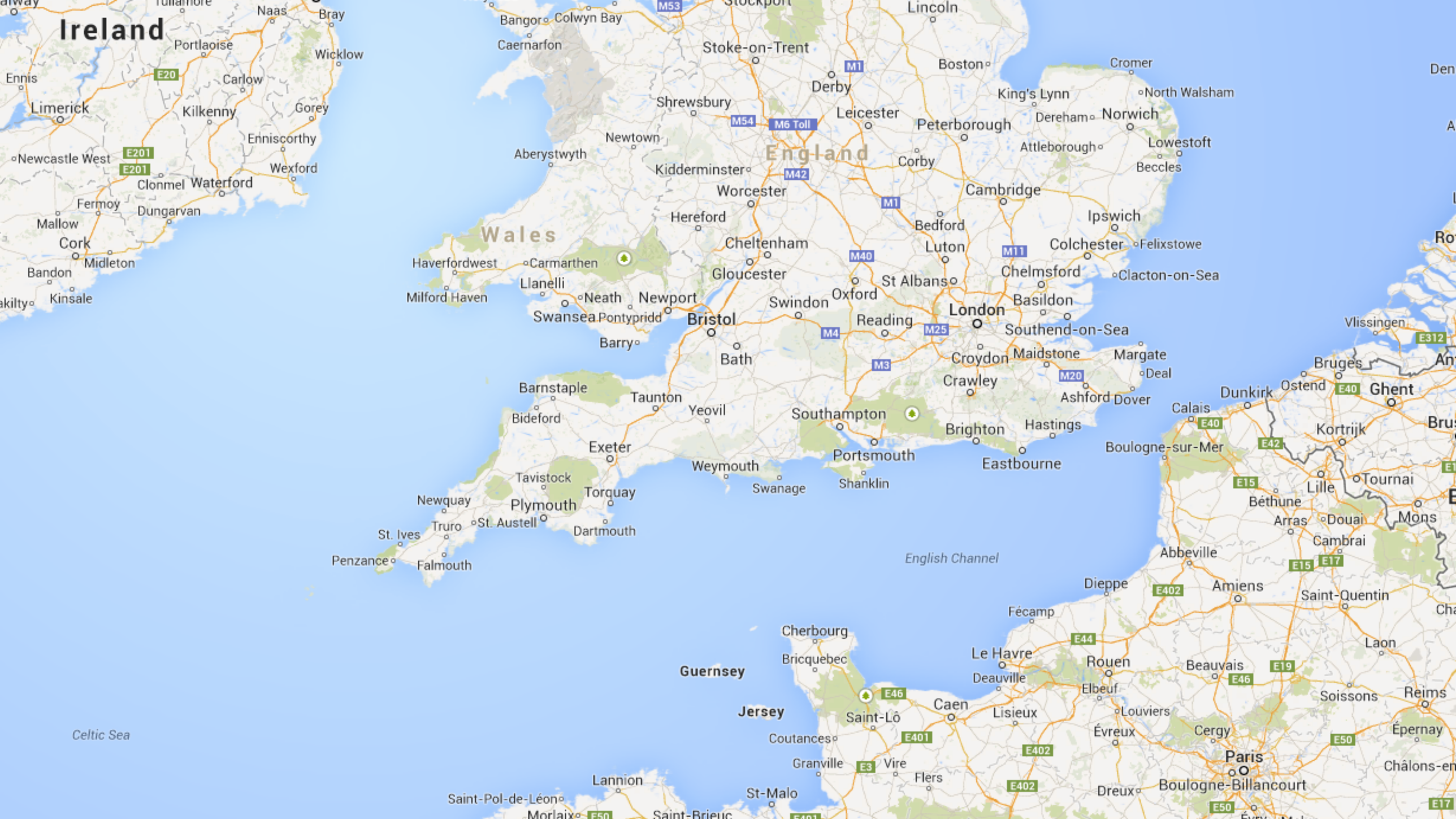
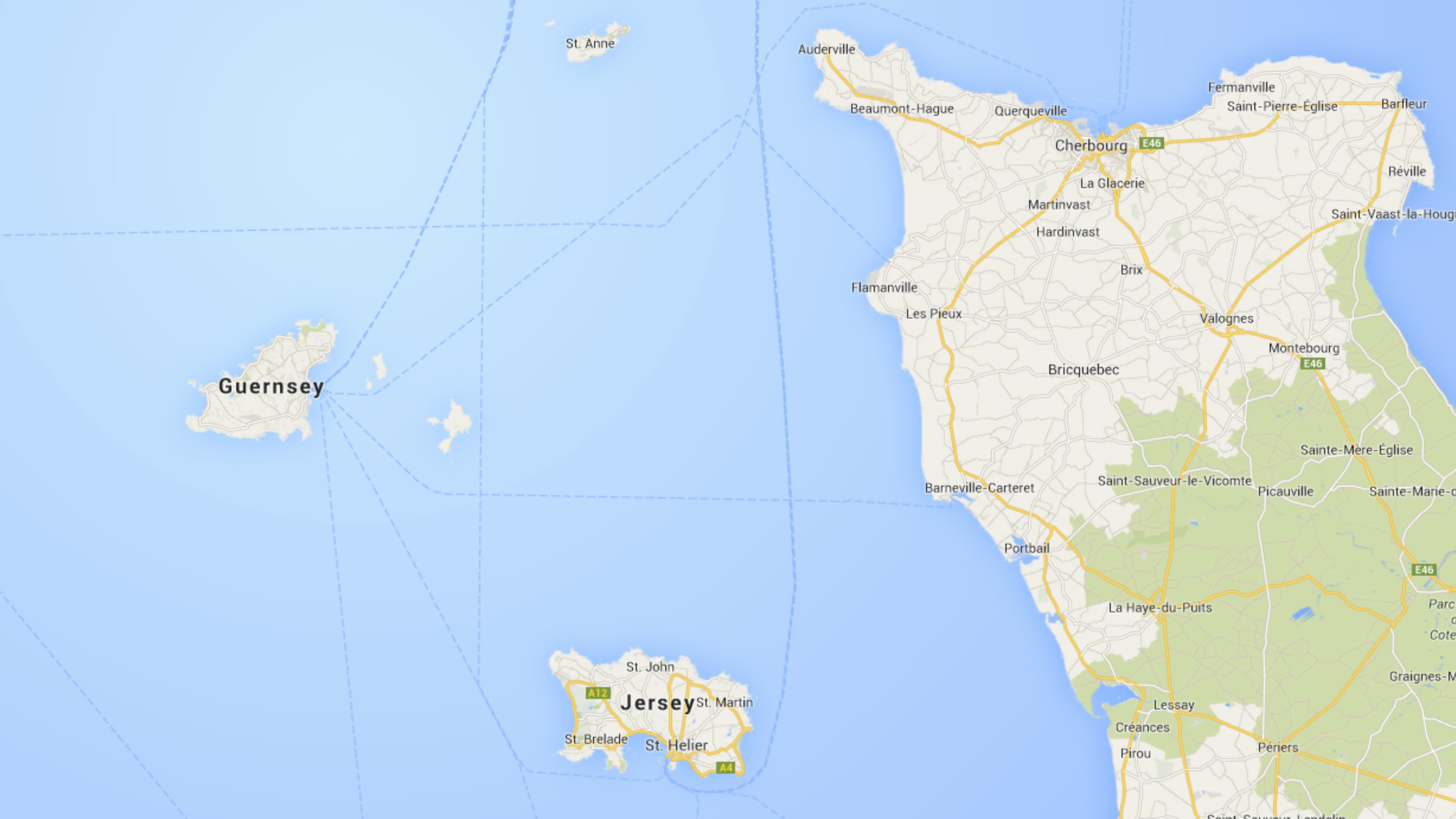
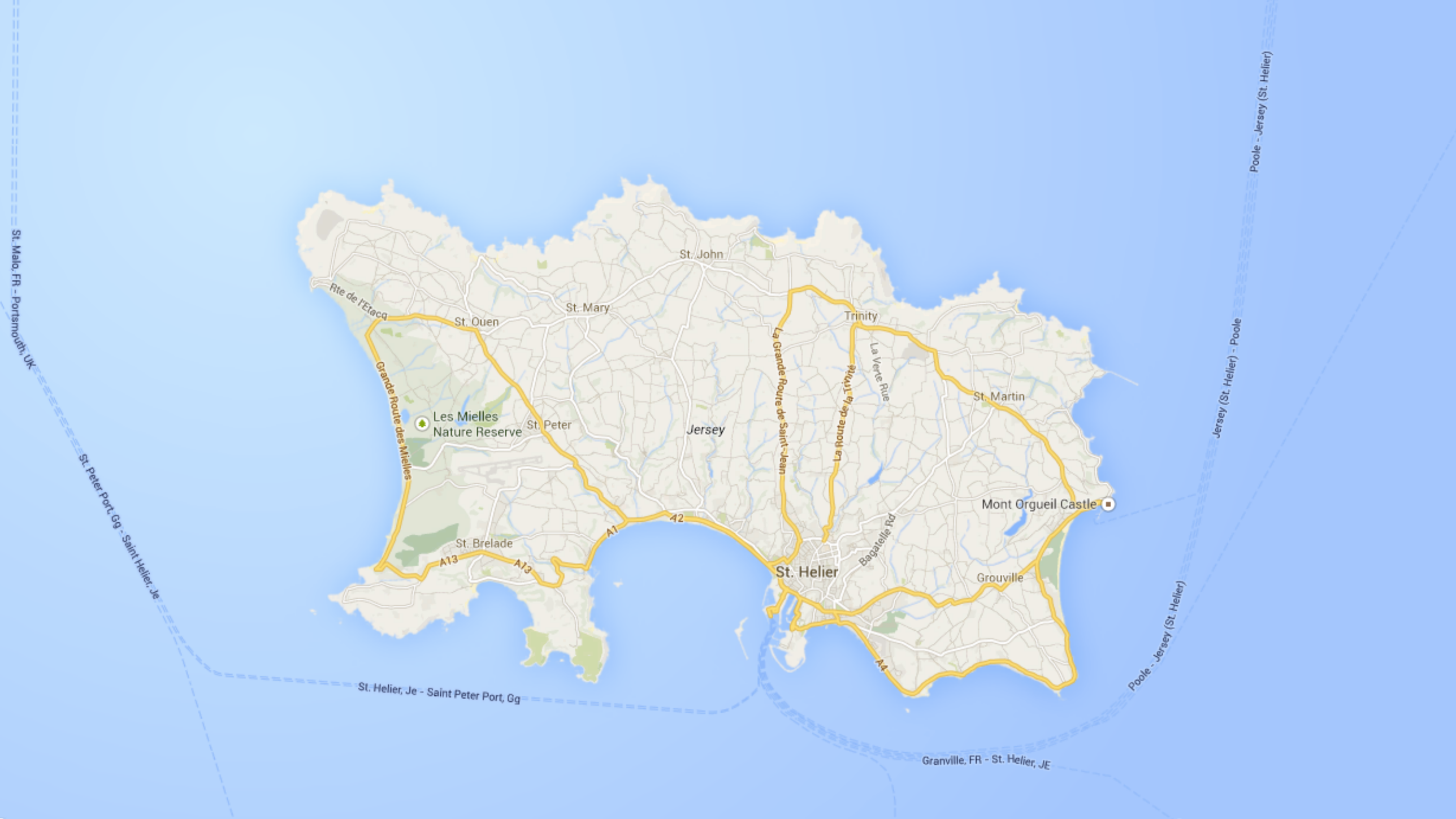
# Visualising software architecture with the C4 model

@simonbrown

North Sea
Perth
Kirkcaldy
Glasgow
Kilmarnock
Ayr
United Kingdom
Londonderry / Derry
Northern Ireland
Bangor
Sligo
Isle of Man
Dundalk
Castlebar
Drogheda
Dublin
Galway
Ireland
Naas
Bray
Limerick
Carlow
Kilkenny
Tralee
Clonmel
Cork
Waterford
Middlesbrough
Blackpool
Leeds
Hull
Liverpool
Sheffield
Stoke-on-Trent
Nottingham
Derby
Norwich
Great Yarmouth
England
Ipswich
Wales
Headington
Swansea
Bristol
London
Leigh-on-Sea
Cardiff
Reading
Barnstaple
Southampton
Exeter
Portsmouth
Bournemouth
Plymouth
Newquay
St. Ives
Truro
English Channel
Celtic Sea
Guernsey
Jersey
Le Havre
Caen
Rouen
Amiens
Saint-Quentin
Reims
Paris
Évry
Lannion
St-Malo
Brest
Saint-Brieuc
Rennes
Quimper
Le Mans
Orleans
Lorient
Vannes
Angers
Tours
Nantes
Saint-Nazaire
Troyes
Dijon
Besançon
Viborg
Aarhus
Silkeborg
Horsens
Esbjerg
Denmark
Odense
Helsingborg
Kristianstad
Karlskrona
Copenhagen
Malmö
Flensburg
Kiel
Heide
Stralsund
Rostock
Swinoujscie
Anklam
Gryfice
Kosza
Aurich
Bremerhaven
Hamburg
Schwerin
Neubrandenburg
Szczecin
Bergedorf
Groningen
Bremen
Stargard Szczecinski
Den Helder
Gorzów Wielkopolski
Berlin
Amsterdam
Hanover
The Hague
Netherlands
Bielefeld
Brunswick
Magdeburg
Dessau
Zielona Gora
Münster
Kernstadt
Cottbus
Woensel-Noord
Leipzig
Bruges
Antwerp
Dusseldorf
Nord
Dresden
Görlitz
Brussels
Germany
Chemnitz
Liberec
Lille
Cologne
Gera
Ústí nad Labem-město
Belgium
Frankfurt
Prague
Wiesbaden
Würzburg
Plzen
Luxembourg
Mannheim
Nuremberg
Saarbrücken
Regensburg
Karlsruhe
Ceske Budejovice
Metz
Stuttgart
Ingolstadt
Strasbourg
Landshut
Nancy
Ulm
Linz
Munich
Freiburg
Salzburg
Wels
Konstanz
Basel
Zurich
Innsbruck
Austria


Ireland
Portlaoise
Naas
Bray
Wicklow
Ennis
Carlow
Limerick
Kilkenny
Gorey
Enniscorthy
Newcastle West
Wexford
Clonmel
Waterford
Fermoy
Dungarvan
Mallow
Cork
Midleton
Bandon
Kinsale
Bangor
Colwyn Bay
Caernarfon
Stoke-on-Trent
Lincoln
Boston
Cromer
Derby
King's Lynn
North Walsham
Shrewsbury
Leicester
Peterborough
Dereham
Norwich
Newtown
Aberystwyth
Attleborough
Lowestoft
Kidderminster
Corby
Beccles
Worcester
Cambridge
Hereford
Bedford
Ipswich
Wales
Cheltenham
Luton
Colchester
Felixstowe
Haverfordwest
Carmarthen
Gloucester
St Albans
Chelmsford
Clacton-on-Sea
Llanelli
Milford Haven
Neath
Newport
Oxford
Basildon
Swansea
Pontypridd
Swindon
London
Bristol
Reading
Southend-on-Sea
Barry
Bath
Croydon
Maidstone
Margate
Deal
Crawley
Ashford
Dover
Barnstaple
Taunton
Yeovil
Southampton
Bideford
Brighton
Hastings
Exeter
Portsmouth
Eastbourne
Weymouth
Tavistock
Swanage
Shanklin
Torquay
Newquay
Plymouth
Truro
St. Austell
St. Ives
Dartmouth
Penzance
Falmouth
English Channel
Vlissingen
Bruges
Ostend
Ghent
Dunkirk
Calais
Kortrijk
Boulogne-sur-Mer
Tournai
Lille
Béthune
Arras
Douai
Mons
Cambrai
Abbeville
Amiens
Dieppe
Saint-Quentin
Fécamp
Cherbourg
Laon
Le Havre
Rouen
Beauvais
Guernsey
Bricquebec
Deauville
Elbeuf
Soissons
Reims
Jersey
Saint-Lô
Caen
Lisieux
Louviers
Évreux
Cergy
Épernay
Celtic Sea
Coutances
Granville
Vire
Paris
Châlons-en
Lannion
Flers
Saint-Pol-de-Léon
St-Malo
Dreux
Boulogne-Billancourt
Morlaix
Saint-Brieuc
M53
M1
M54
M6 Toll
M42
M40
M11
M4
M25
M3
M20
E20
E201
E312
E40
E42
E15
E17
E402
E44
E19
E46
E401
E3
E50


St. Anne
Auderville
Beaumont-Hague
Querqueville
Fermanville
Saint-Pierre-Église
Barfleur
Cherbourg
E46
Réville
La Glacerie
Martinvast
Saint-Vaast-la-Hougue
Hardinvast
Brix
Flamanville
Les Pieux
Valognes
Montebourg
E46
Bricquebec
Guernsey
Sainte-Mère-Église
Saint-Sauveur-le-Vicomte
Barneville-Carteret
Picauville
Sainte-Marie-d
Portbail
E46
La Haye-du-Puits
Parc
Cote
Graignes-M
St. John
A12
Jersey
St. Martin
Lessay
Créances
St. Brelade
St. Helier
Pirou
Périers
A4


St. John
Rte de l'Etacq
St. Ouen
St. Mary
Trinity
La Grande Route de Saint-Jean
La Route de la Trinité
La Verte Rue
St. Martin
Grande Route des Mielles
Les Mielles
Nature Reserve
St. Peter
Jersey
Mont Orgueil Castle
A2
A1
St. Brelade
A13
A13
St. Helier
Bagatelle Rd
Grouville
A4
St. Malo, FR - Portsmouth, UK
St Peter Port, Gg - Saint Helier, Je
St. Helier, Je - Saint Peter Port, Gg
Granville, FR - St. Helier, JE
Poole - Jersey (St. Helier)
Jersey (St. Helier) - Poole
Poole - Jersey (St. Helier)

https://leanpub.com/visualising-software-architecture**/c/...**

# What is software architecture?

# Structure

The definition of software in terms
of its building blocks and their interactions

# Vision

The process of architecting;
making decisions based upon business goals,
requirements and constraints,
plus being able to communicate this to a team

# Enterprise Architecture

Structure and strategy across people, process and technology

## System Architecture

High-level structure of a software system
(software and infrastructure)

## Application Architecture

The internal structure of an application

As a noun, **design is the named structure** or behaviour of a system whose presence resolves … a force on that system. A design thus represents one point in a potential decision space.

Grady Booch

All architecture is design, but
**not all design is architecture**.

Grady Booch

Architecture represents the **significant decisions**, where significance is measured by **cost of change**.

Grady Booch

**Draw** one or more software architecture diagrams to describe the system/project you are currently working on

**60 minutes**

**Design** a software solution for the ”Financial Risk System”, and **draw** one or more architecture diagrams to describe your solution

**90 minutes**

Did you find anything about this exercise **challenging**?

## Challenging?

Level of detail
- where to stop

Who is the audience / different backgrounds

Implementation
- easy to get bogged down in detail

Type of diagrams

Notation

Documenting assumptions

## (10) Challenging?

Verifying our own assumptions

Expressing the solution
- communicating it in a clear way
- use of notation
- easy to mix levels of abstraction
- how much detail?

## (7) Challenging

Needed to ask questions / make assumptions

Temptation to focus on detail
- when do we stop?

How much detail?

Talked about more than the diagrams

What notation?
- boxes
- arrows

# **Swap** and **review** your diagrams

Focus on the diagrams rather than the solution itself;
do you understand the notation, colour coding, symbols, etc?

**15 minutes**

# The perfection game

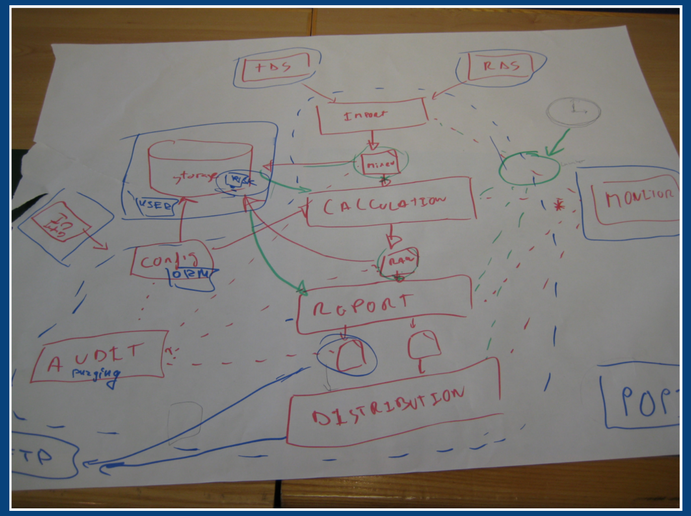
We rate the diagrams... (1-10)

We liked...

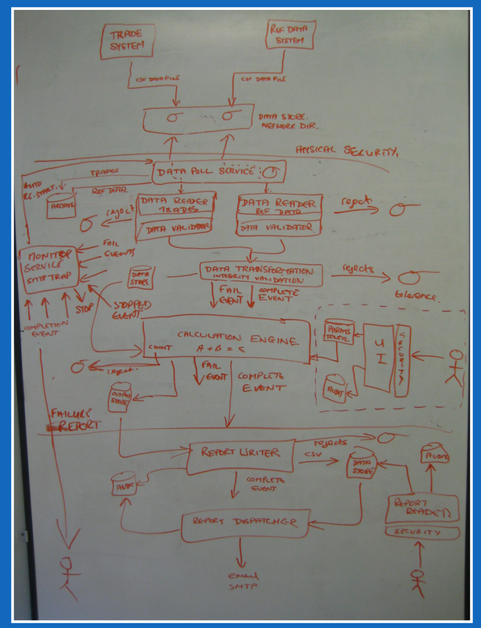
To make the diagrams perfect...

**15 minutes**

1 1 1


TDS
RDS
Import
Mixer
storage
MQ
VSEB
CALCULATION
Raw
Config
OPN
REPORT
AUDIT
Parsing
DISTRIBUTION

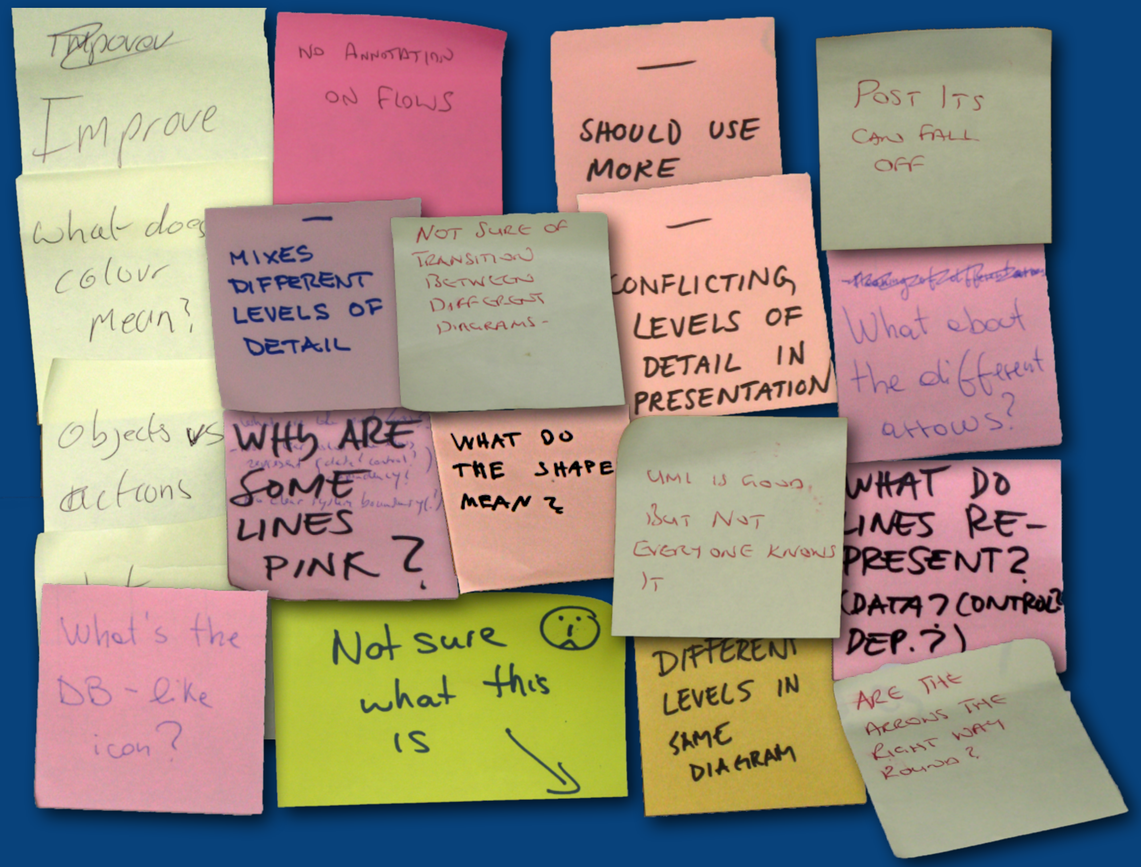
2 2 2

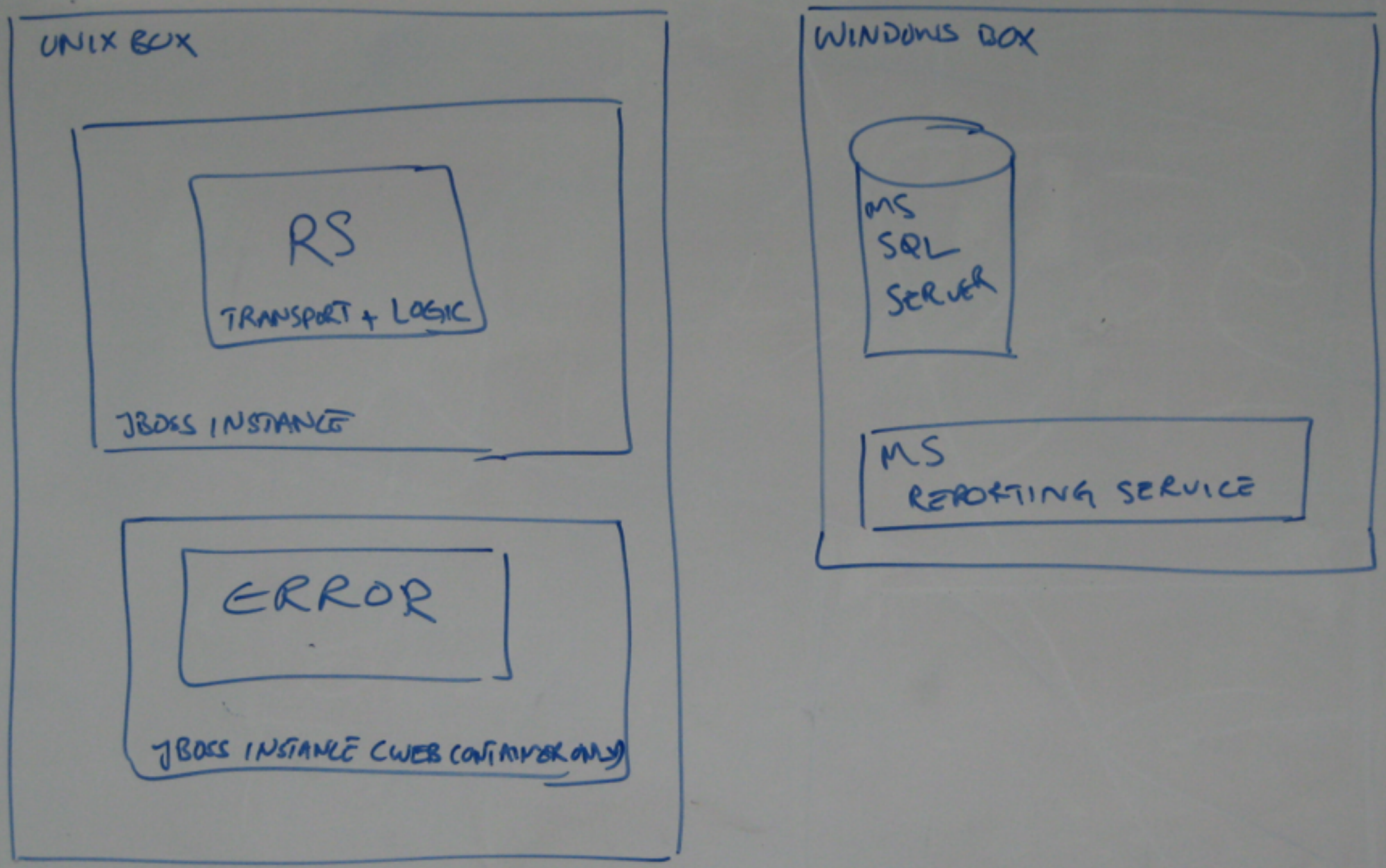

TRADE SYSTEM
REF DATA SYSTEM
DATA SERVE NETWORK DIR.
PHYSICAL SECURITY
DATA PULL SERVICE
DATA READER TRADES
DATA READER REF DATA
DATA VALIDATOR
reports
MONITOR SERVICE SNMP TRAP
DATA TRANSFORMATION INTEGRITY VALIDATION
rejects
balance
STOPPED EVENT
STOP
CALCULATION ENGINE
A + B = C
FAIL EVENT
COMPLETE EVENT
UI
FAILURE REPORT
REPORT WRITER
REPORT DISPATCHER
REPORT BROKER
SECURITY
email SMTP

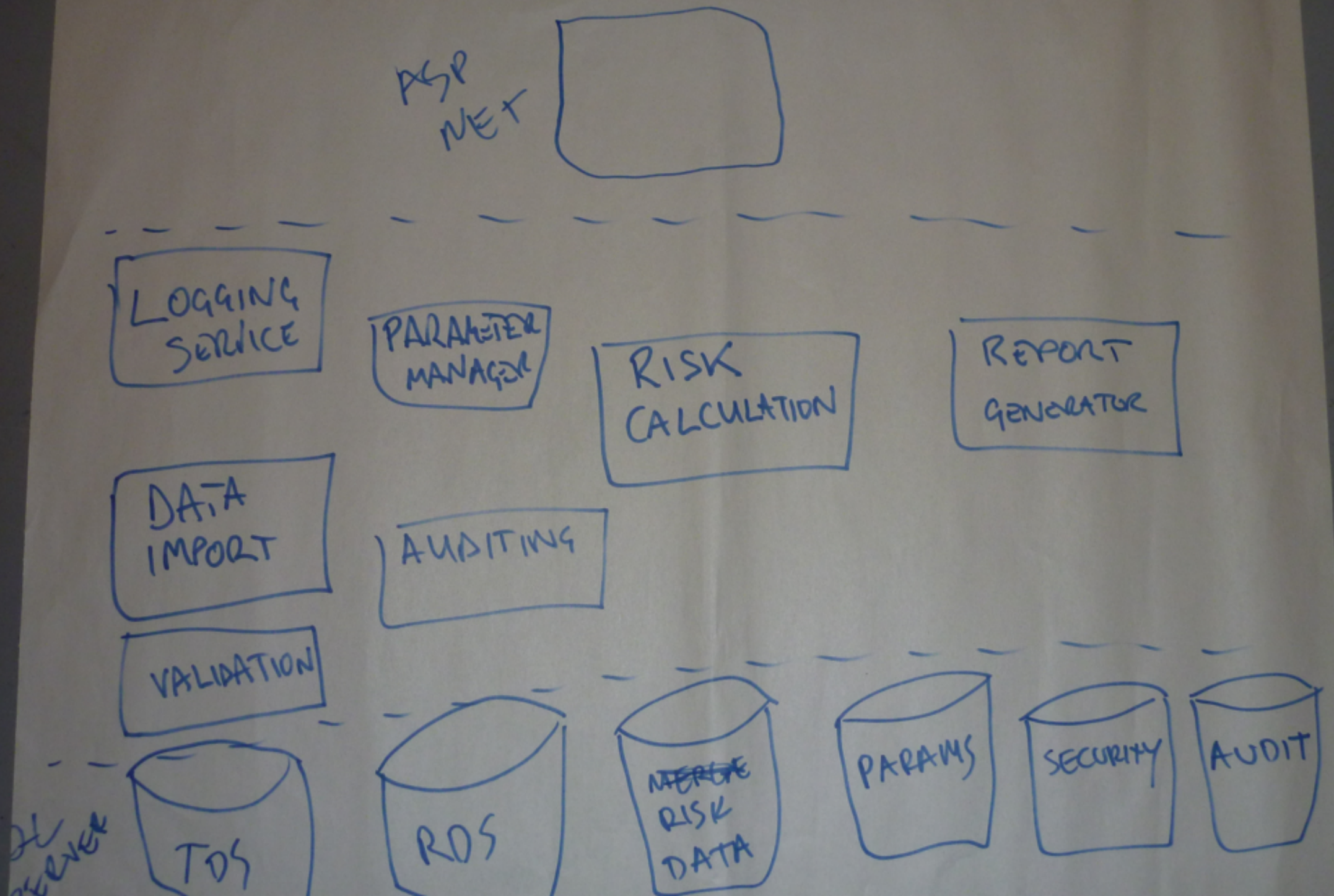
Information is likely
still stuck in your heads

“This doesn’t make sense, but we’ll explain it.”

- What is this shape/symbol?
- What is this line/arrow?
- What do the colours mean?
- What level of abstraction is shown?
- Which diagram do we read first?




UNIX BOX
RS
TRANSPORT + LOGIC
JBOSS INSTANCE
ERROR
JBOSS INSTANCE (WEB CONTAINER ONLY)
WINDOWS BOX
MS
SQL
SERVER
MS
REPORTING SERVICE


ASP NET
LOGGING SERVICE
PARAMETER MANAGER
RISK CALCULATION
REPORT GENERATOR
DATA IMPORT
AUDITING
VALIDATION
TDS
RDS
~~MERGE~~ RISK DATA
PARAMS
SECURITY
AUDIT

# FUNCTIONAL VIEW

File Retriever

Scheduler

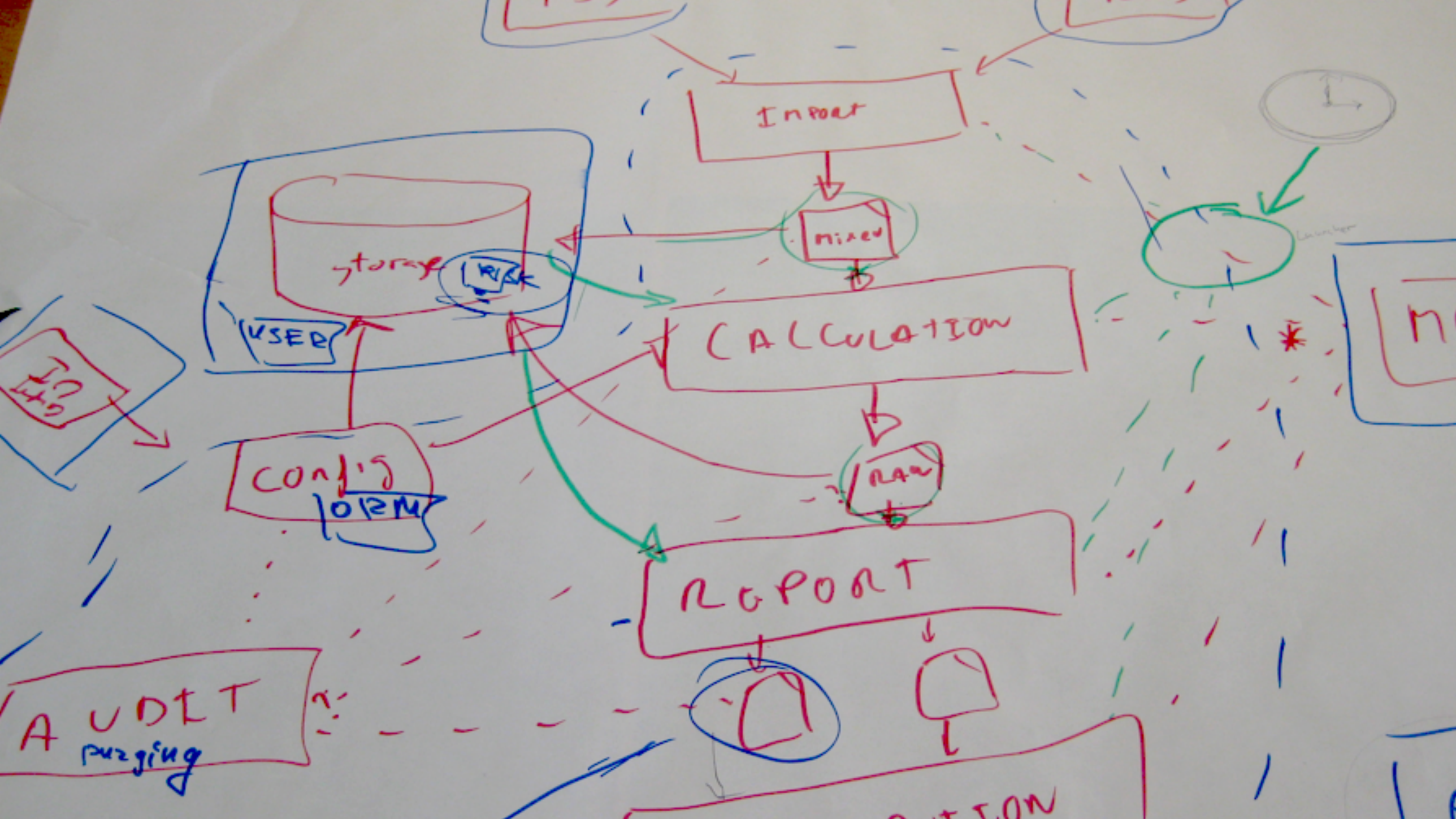
Auditing

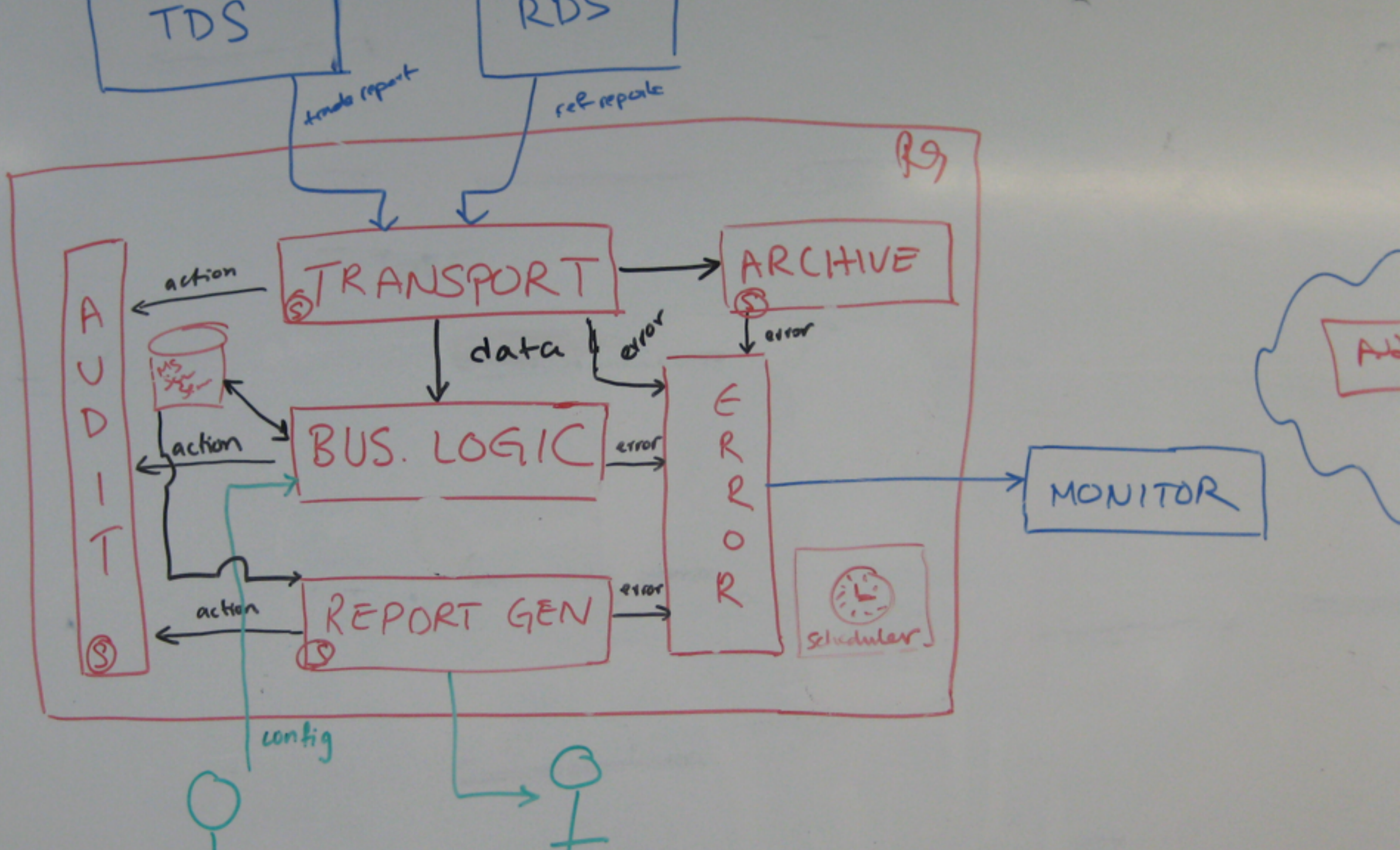
Reference Archiver

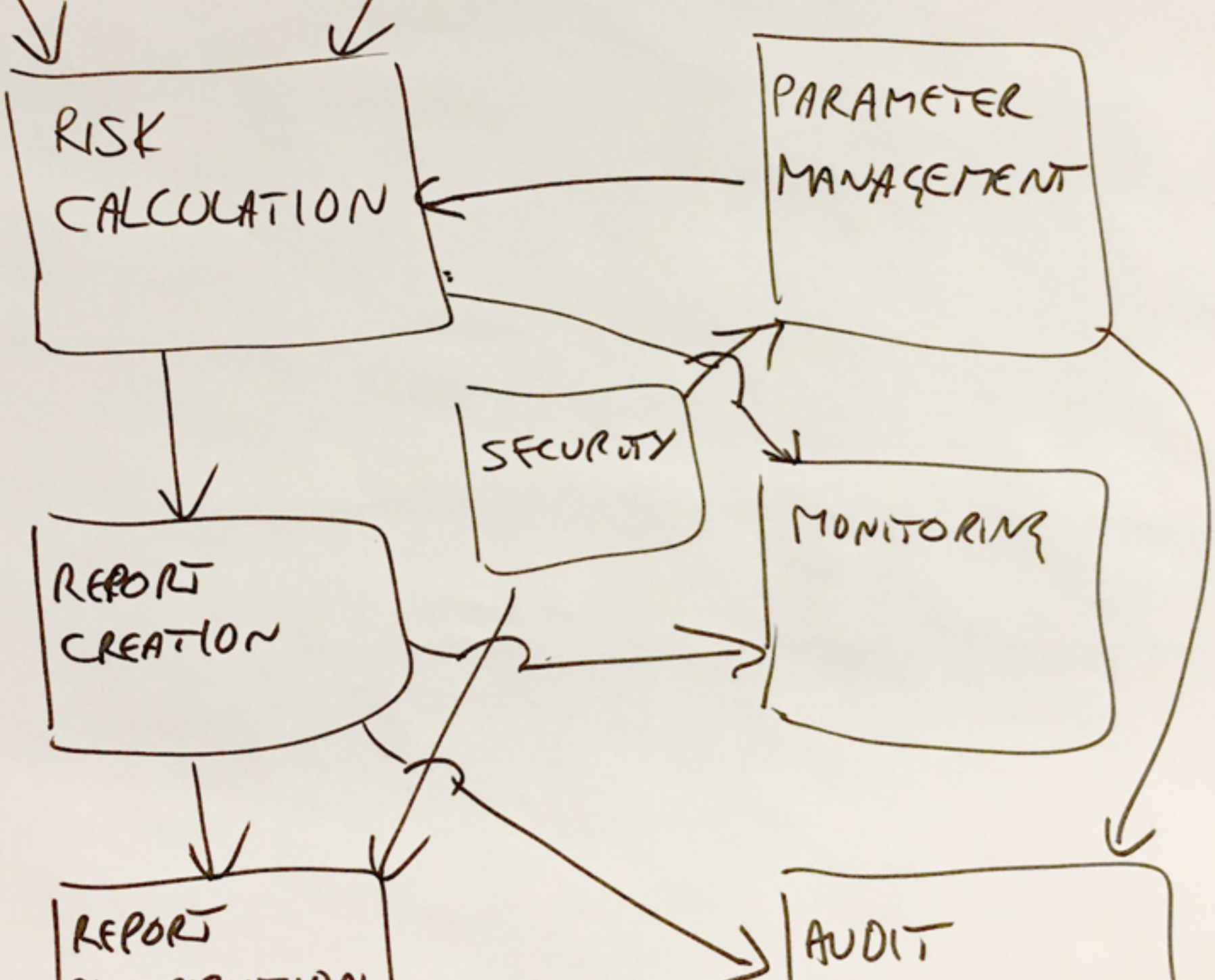
Risk Assesment Processor

Risk Parameter Configuration

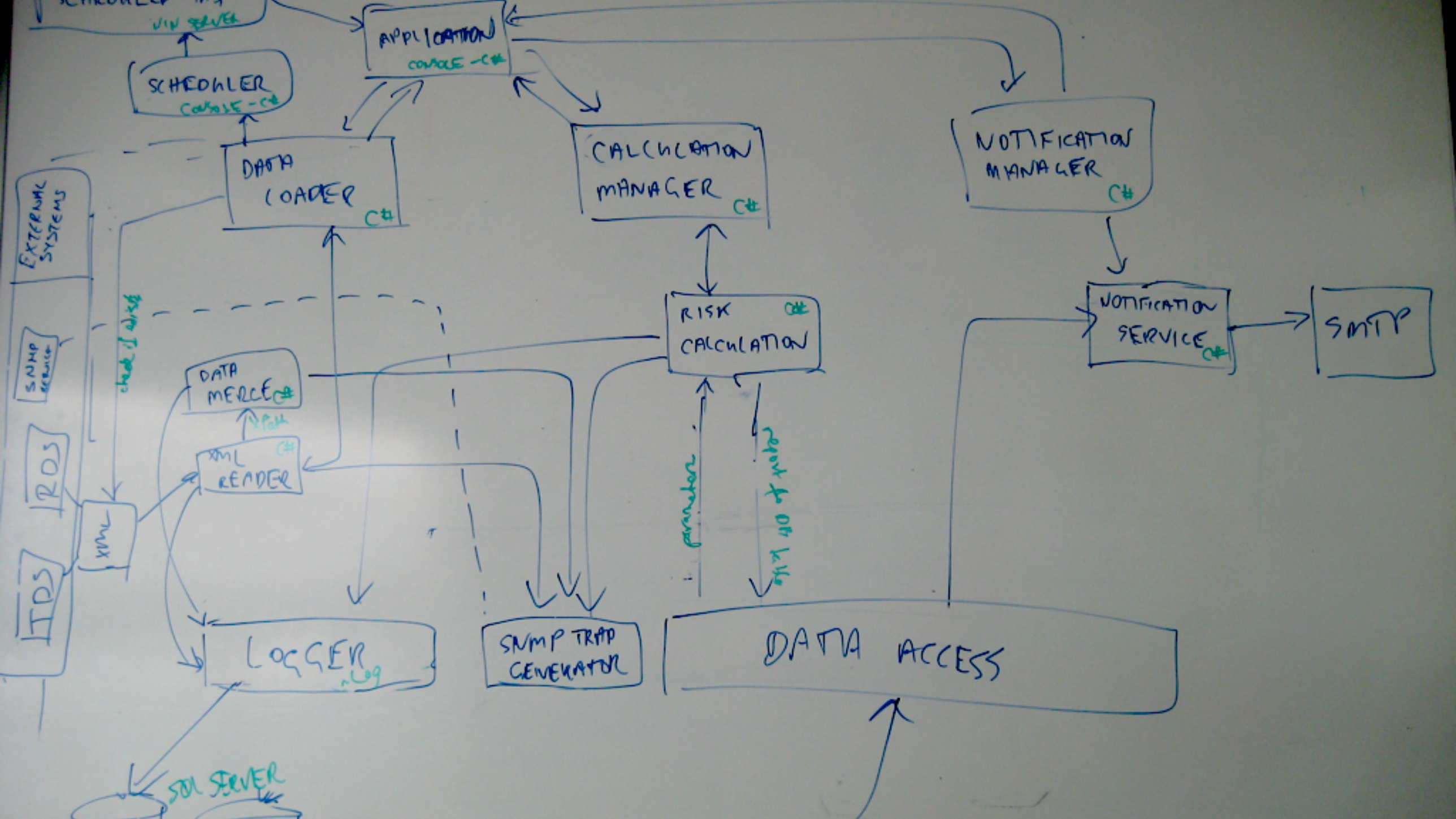
Report

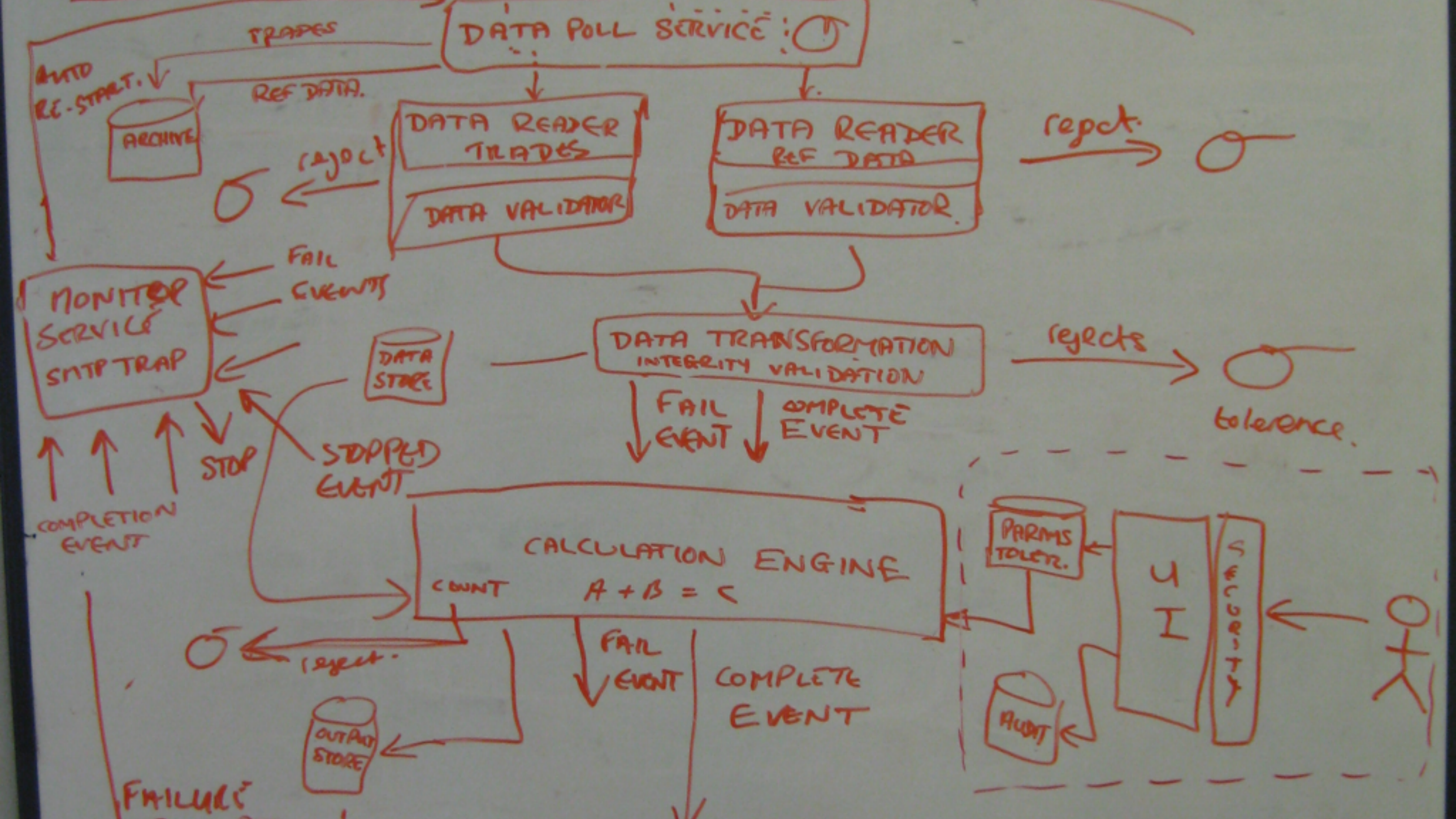

Import
mixer
storage
RISK
USER
CALCULATION
GTI
config
ORM
RAW
REPORT
AUDIT
purging

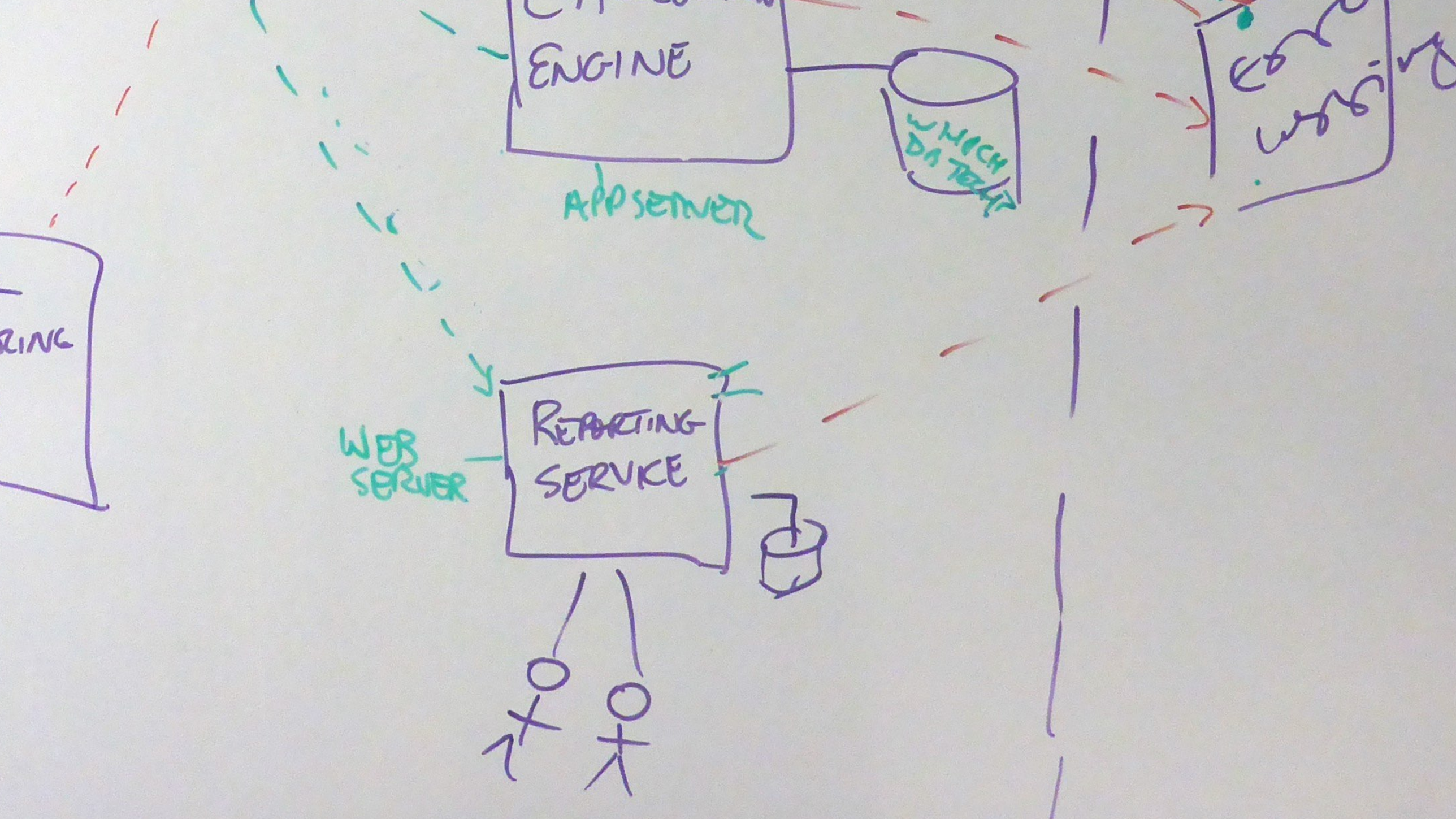

TDS
RDS
trade report
ref repo
RG
TRANSPORT
ARCHIVE
AUDIT
action
data
error
error
BUS. LOGIC
action
error
ERROR
MONITOR
REPORT GEN
action
error
Scheduler
config


RISK CALCULATION
PARAMETER MANAGEMENT
SECURITY
MONITORING
REPORT CREATION
REPORT
AUDIT

2
SERVICE/BATCH
IMPORT
JOIN
CALC
EXPORT/STORE
REPORT
SNMP
CENTRAL
MONITORING
SERVICE
3
WEBAPP
•GET REPORT
•EDIT
RISK RULES
•AUTH
•AUDIT
RISK RULES


VIN SERVER
APPLICATION
CONSOLE - C#
SCHEDULER
CONSOLE - C#
DATA LOADER
C#
CALCULATION MANAGER
C#
NOTIFICATION MANAGER
C#
EXTERNAL SYSTEMS
SNMP SERVICE
ROS
IDS
check if alive
DATA MERGE C#
XPath
XML READER
C#
XML
RISK CALCULATION
C#
NOTIFICATION SERVICE
C#
SMTP
parameters
output of risk
LOGGER
.log
SNMP TRAP GENERATOR
DATA ACCESS
SQL SERVER


DATA POLL SERVICE
TRADES
AUTO RE-START.
REF DATA.
ARCHIVE
DATA READER TRADES
DATA VALIDATOR
reject
DATA READER REF DATA
DATA VALIDATOR
reject
MONITOR SERVICE SMTP TRAP
FAIL EVENTS
DATA STORE
DATA TRANSFORMATION INTEGRITY VALIDATION
rejects
tolerence.
FAIL EVENT
COMPLETE EVENT
STOP
STOPPED EVENT
COMPLETION EVENT
CALCULATION ENGINE
COUNT
A + B = C
reject.
FAIL EVENT
COMPLETE EVENT
OUTPUT STORE
FAILURE
PARAMS TOLER.
UI
SECURITY
AUDIT


ENGINE
APP SERVER
WEB SERVER
REPORTING SERVICE

Report Monitor
(other machine)
Data Normalizer
Risk ~~Calculator~~ Aggregator
CP Risk Calc
Report Generator
Auditing Service
Report Publisher
Report Repository

Params
Calcs
EH?
Floor
App Log
Date
- Risk Cell
- calcs
- Par
Batch
Params_Distrib
B_id:
Date id

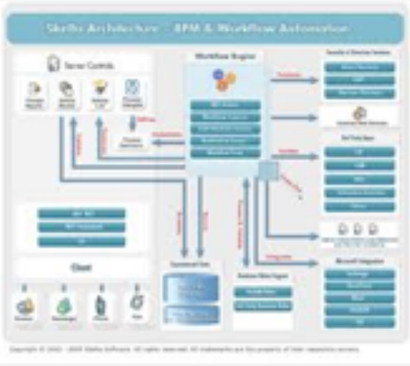
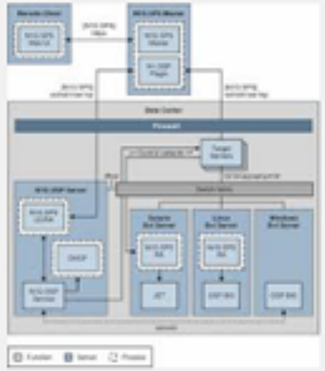
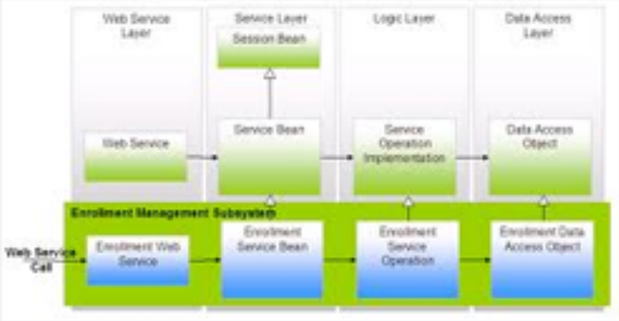
Google software architecture diagram

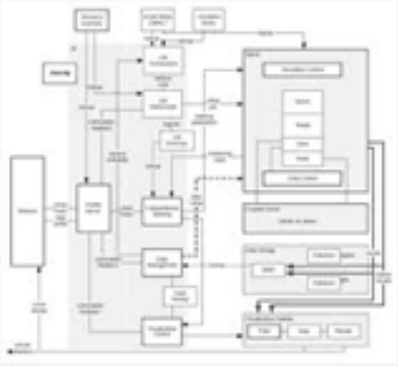
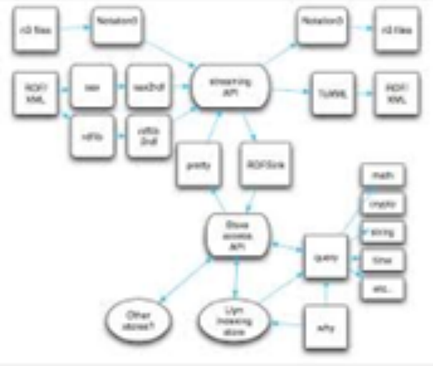
Sign in

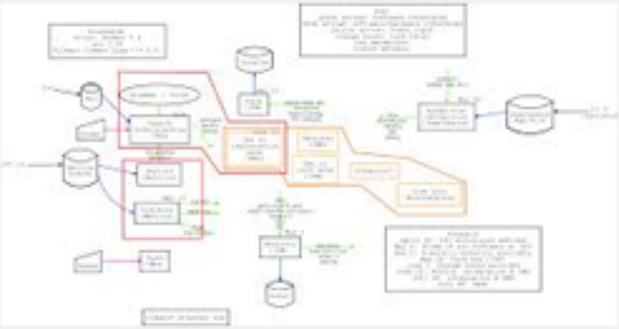
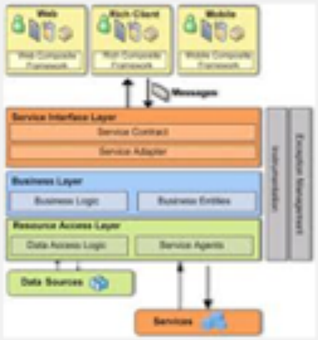
Web Images Videos Shopping News More Search tools

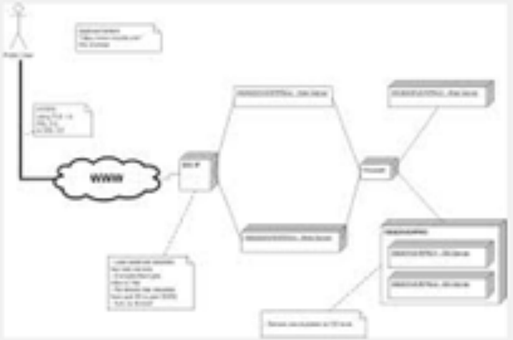
SafeSearch

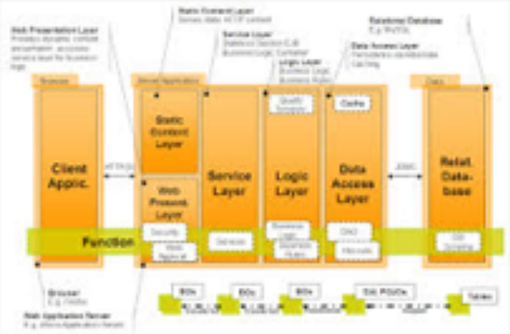
Cookies help us deliver our services. By using our services, you agree to our use of cookies.
Learn more Got It

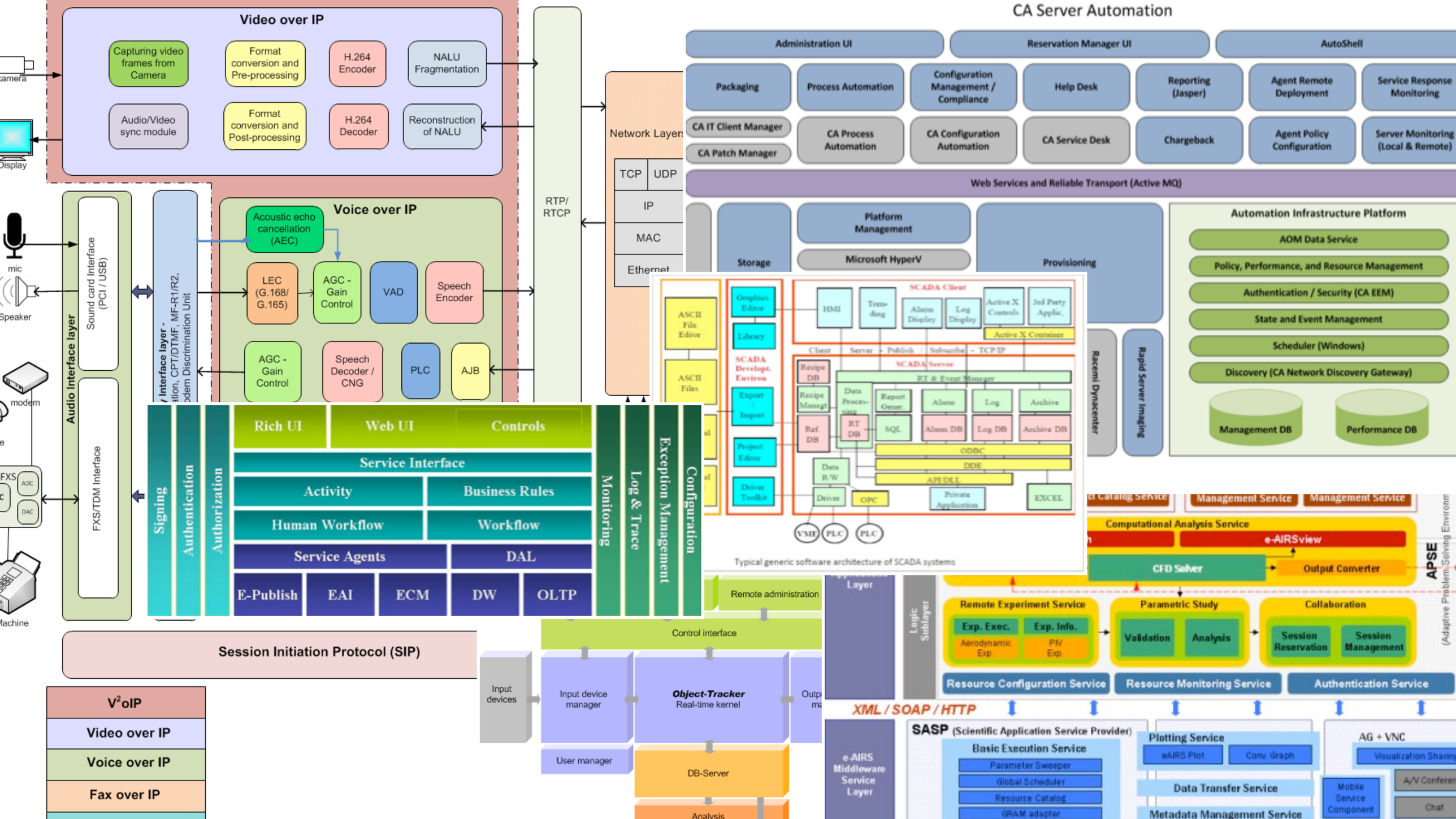









Video over IP
Capturing video frames from Camera
Format conversion and Pre-processing
H.264 Encoder
NALU Fragmentation
Audio/Video sync module
Format conversion and Post-processing
H.264 Decoder
Reconstruction of NALU
camera
Display
Voice over IP
Acoustic echo cancellation (AEC)
LEC (G.168/ G.165)
AGC - Gain Control
VAD
Speech Encoder
AGC - Gain Control
Speech Decoder / CNG
PLC
AJB
mic
Speaker
modem
Audio Interface layer
Sound card Interface (PCI / USB)
FXS/TDM Interface
Interface layer - tion, CPT/DTMF, MF-R1/R2, odem Discrimination Unit
RTP/ RTCP
Network Layers
TCP
UDP
IP
MAC
Ethernet
Signing
Authentication
Authorization
Rich UI
Web UI
Controls
Service Interface
Activity
Business Rules
Human Workflow
Workflow
Service Agents
DAL
E-Publish
EAI
ECM
DW
OLTP
Monitoring
Log & Trace
Exception Management
Configuration
Session Initiation Protocol (SIP)
V²oIP
Video over IP
Voice over IP
Fax over IP
CA Server Automation
Administration UI
Reservation Manager UI
AutoShell
Packaging
Process Automation
Configuration Management / Compliance
Help Desk
Reporting (Jasper)
Agent Remote Deployment
Service Response Monitoring
CA IT Client Manager
CA Process Automation
CA Configuration Automation
CA Service Desk
Chargeback
Agent Policy Configuration
Server Monitoring (Local & Remote)
CA Patch Manager
Web Services and Reliable Transport (Active MQ)
Storage
Platform Management
Microsoft HyperV
Provisioning
Automation Infrastructure Platform
AOM Data Service
Policy, Performance, and Resource Management
Authentication / Security (CA EEM)
State and Event Management
Scheduler (Windows)
Discovery (CA Network Discovery Gateway)
Management DB
Performance DB
Racemi Dynacenter
Rapid Server Imaging
ASCII File Editor
ASCII Files
Graphics Editor
Library
SCADA Developt. Environ
Export / Import
Project Editor
Driver Toolkit
SCADA Client
HMI
Trending
Alarm Display
Log Display
Active X Controls
3rd Party Applic.
Active X Container
Client - Server - Publish - Subscribe - TCP/IP
SCADA Server
Recipe DB
Recipe Managt
Ref. DB
Data Process-ing
RT & Event Manager
RT DB
Report Gener.
SQL
Alarm
Log
Archive
Alarm DB
Log DB
Archive DB
ODBC
DDE
API/DLL
Data R/W
Driver
OPC
Private Application
EXCEL
VME
PLC
PLC
Typical generic software architecture of SCADA systems
Remote administration
Control interface
Input devices
Input device manager
Object-Tracker
Real-time kernel
User manager
DB-Server
Analysis
Management Service
Management Service
Computational Analysis Service
e-AIRSview
CFD Solver
Output Converter
APSE
(Adaptive Problem Solving Environm
Remote Experiment Service
Exp. Exec.
Exp. Info.
Aerodynamic Exp.
PIV Exp.
Parametric Study
Validation
Analysis
Collaboration
Session Reservation
Session Management
Resource Configuration Service
Resource Monitoring Service
Authentication Service
Logic Sublayer
XML / SOAP / HTTP
e-AIRS Middleware Service Layer
SASP (Scientific Application Service Provider)
Basic Execution Service
Parameter Sweeper
Global Scheduler
Resource Catalog
SRAM adapter
Plotting Service
eAIRS Plot
Conv. Graph
Data Transfer Service
Metadata Management Service
AG + VNC
Visualization Sharing
Mobile Service Component
A/V Conferen
Chat

Moving fast in the same direction
as a team requires

**good communication**

I've run diagramming workshops

in **30+ countries**

for **10,000+ people**

Software architects

**struggle to communicate**

software architecture

Do **you** use UML?

In my experience, optimistically,

# 1 out of 10 people use UML

Who are the **stakeholders** that you need to communicate software architecture to; what **information** do they need?

There are many **different audiences** for diagrams and documentation, all with **different interests**

(software architects, software developers, operations and support staff, testers, Product Owners, project managers, Scrum Masters, users, management, business sponsors, potential customers, potential investors, ...)

The primary use for diagrams and documentation is **communication** and **learning**

Software architecture diagrams, when connected to the code, are also a fantastic tool for **architectural improvement**

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

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.

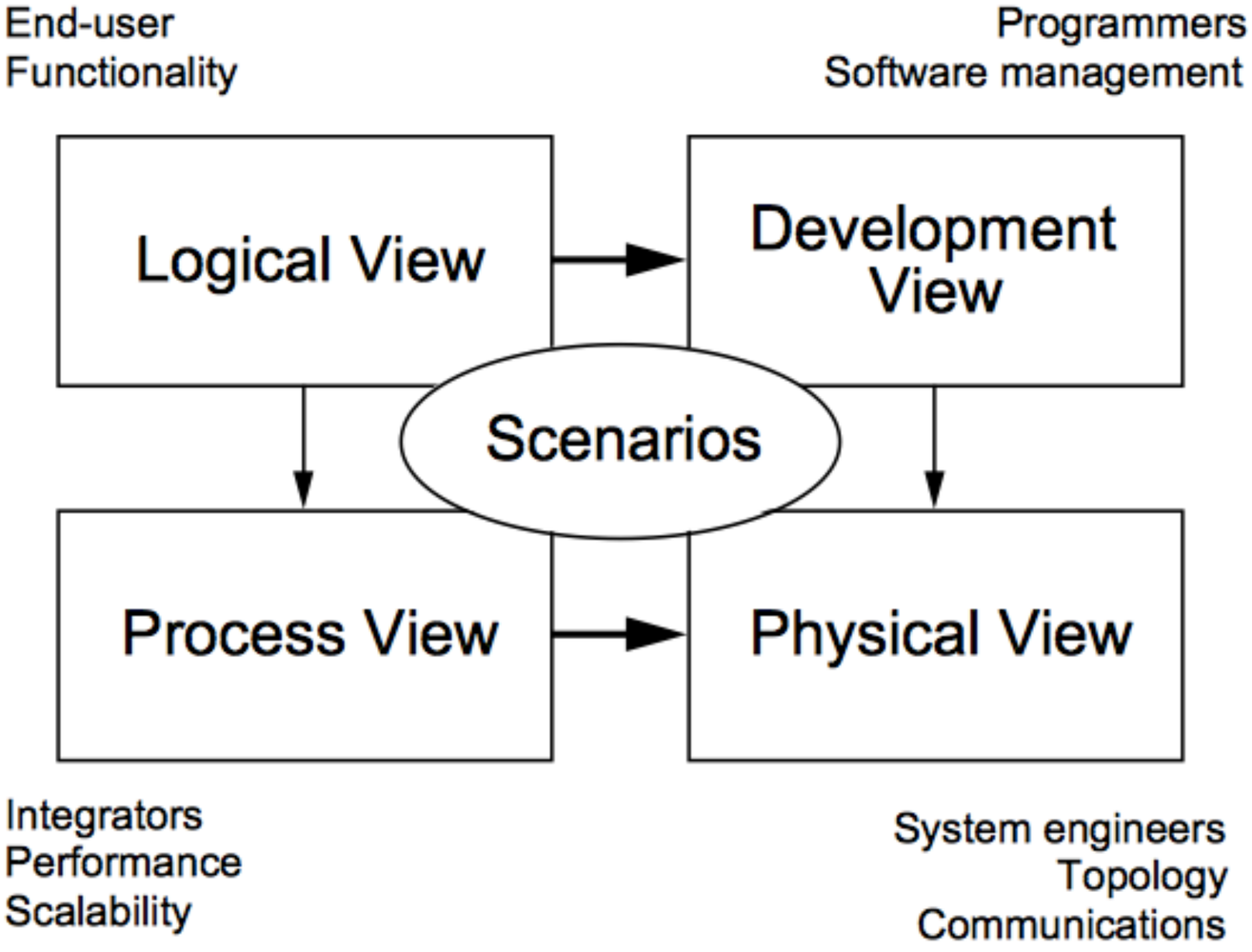
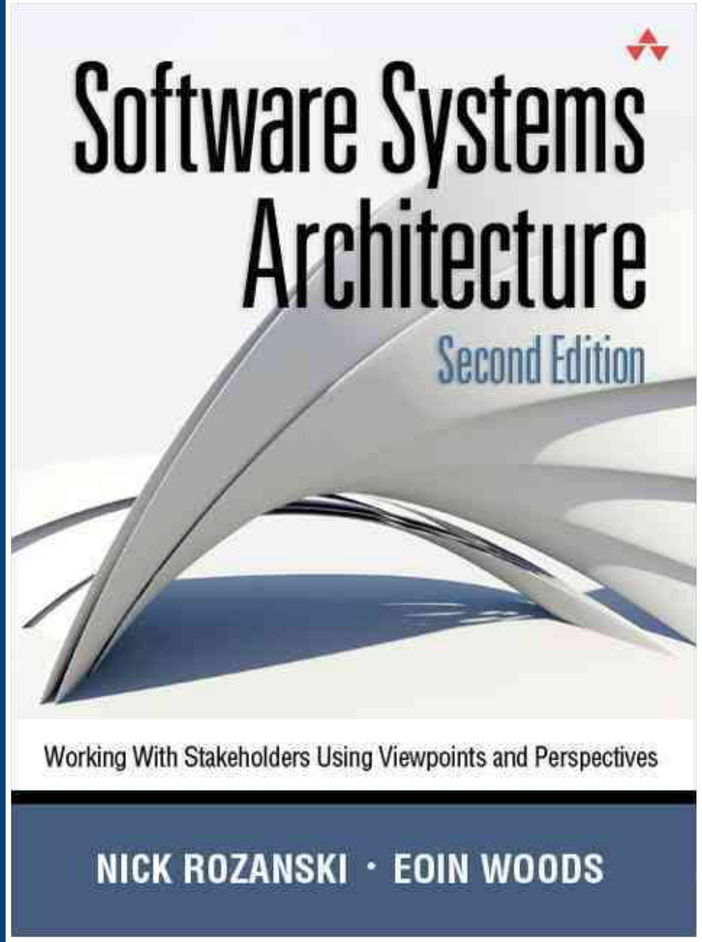
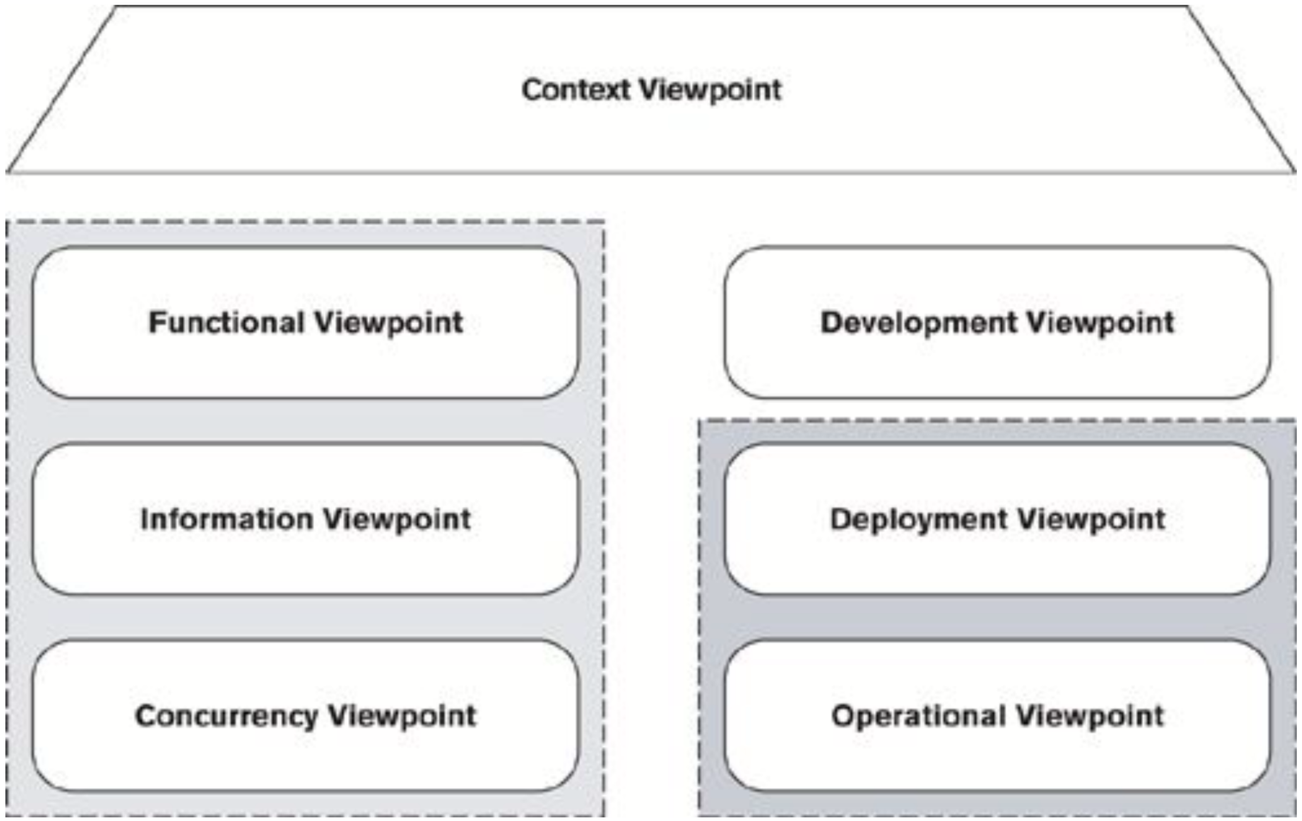



Figure 1 — The "4+1" view model

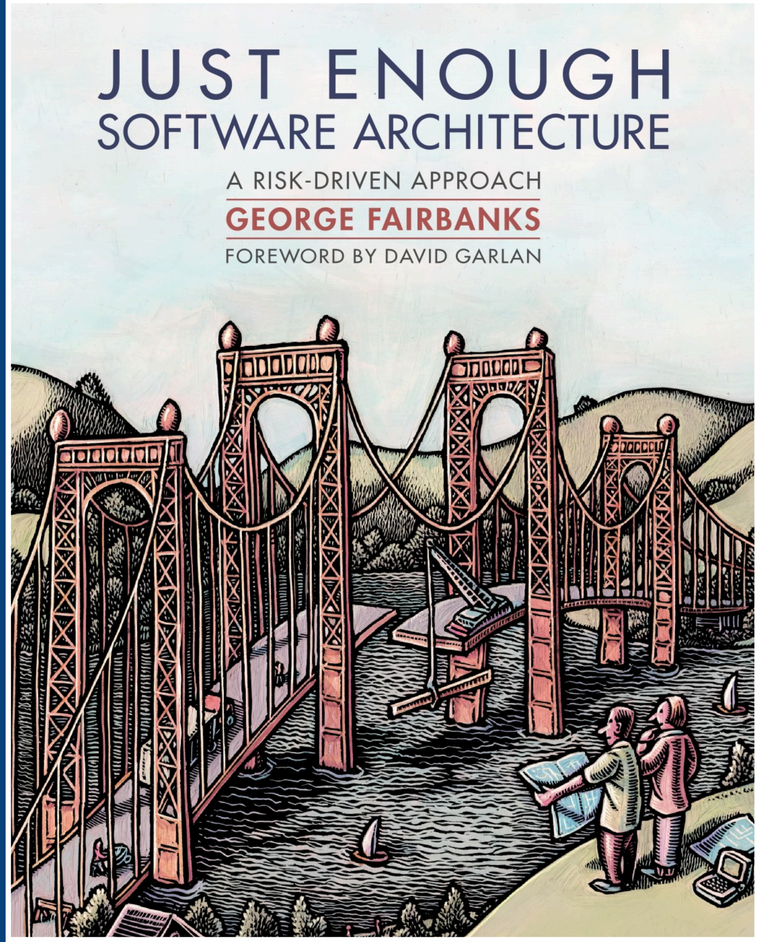



# “Viewpoints and Perspectives”

Why is there a separation between the **logical** and **development** views?

“Our architecture diagrams don’t match the code.”




**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.

# “model-code gap”

# Software Reflexion Models: Bridging the Gap between Source and High-Level Models*

Gail C. Murphy and David Notkin

Dept. of Computer Science & Engineering
University of Washington
Box 352350
Seattle WA, USA 98195-2350
{gmurphy, notkin}@cs.washington.edu

Kevin Sullivan

Dept. of Computer Science
University of Virginia
Charlottesville VA, USA 22903
sullivan@cs.virginia.edu

## Abstract

Software engineers often use high-level models (for instance, box and arrow sketches) to reason and communicate about an existing software system. One problem with high-level models is that they are almost always inaccurate with respect to the system's source code. We have developed an approach that helps an engineer use a high-level model of the structure of an existing software system as a lens through which to see a model of that system's source code. In particular, an engineer defines a high-level model and specifies how the model maps to the source. A tool then computes a software reflexion model that shows where the engineer's high-level model agrees with and where it differs from a model of the source.

The paper provides a formal characterization of reflexion models, discusses practical aspects of the approach, and relates experiences of applying the approach and tools to a number of different systems. The illustrative example used in the paper describes the application of reflexion models to NetBSD, an implementation of Unix comprised of 250,000 lines of C code. In only a few hours, an engineer computed several reflexion models that provided him with a useful, global overview of the structure of the NetBSD virtual memory subsystem. The approach has also been applied to aid in the understanding and experimental reengineering of the Microsoft Excel spreadsheet product.

*This research was funded in part by the NSF grant CCR-8858804 and a Canadian NSERC post-graduate scholarship.

[0]Permission to make digital/hard copies of all or part of this material without fee is granted provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication and its date appear, and notice is given that copyright is by permission of the Association for Computing Machinery, Inc. (ACM). To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGSOFT '95 Washington, D.C., USA
©1995 ACM 0-89791-716-2/95/0010...$3.50

## 1 Introduction

Software engineers often think about an existing software system in terms of high-level models. Box and arrow sketches of a system, for instance, are often found on engineers' whiteboards. Although these models are commonly used, reasoning about the system in terms of such models can be dangerous because the models are almost always inaccurate with respect to the system's source.

Current reverse engineering systems derive high-level models from the source code. These derived models are useful because they are, by their very nature, accurate representations of the source. Although accurate, the models created by these reverse engineering systems may differ from the models sketched by engineers; an example of this is reported by Wong et al. [WTMS95].

We have developed an approach, illustrated in Figure 1, that enables an engineer to produce sufficiently accurate high-level models in a different way. The engineer defines a high-level model of interest, extracts a source model (such as a call graph or an inheritance hierarchy) from the source code, and defines a declarative mapping between the two models. A *software reflexion model* is then computed to determine where the engineer's high-level model does and does not agree with the source model.[1] An engineer interprets the reflexion model and, as necessary, modifies the input to iteratively compute additional reflexion models.

[1]The old English spelling differentiates our use of "reflexion" from the field of reflective computing [Smi84].

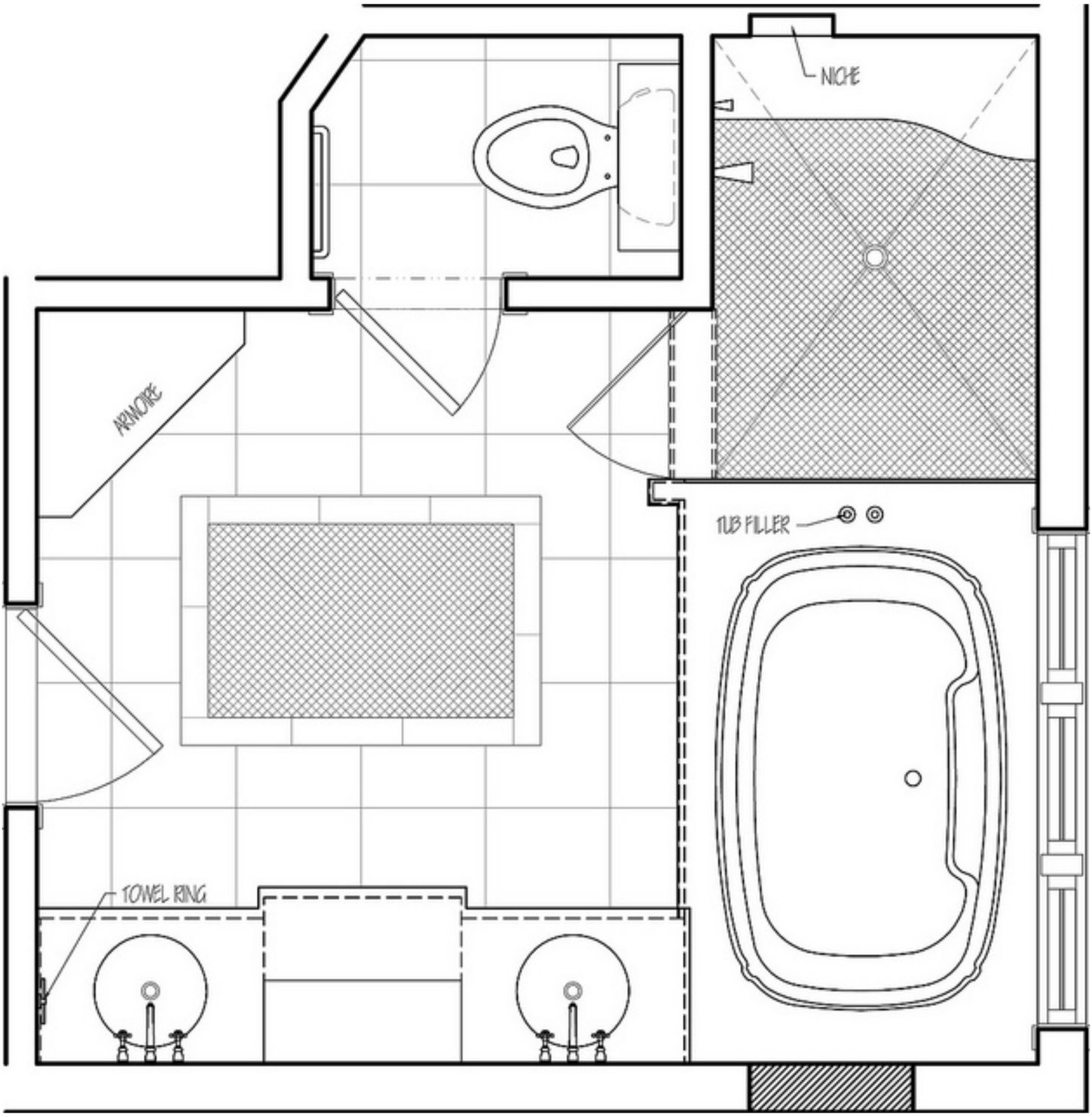
# 1 Introduction

Software engineers often think about an existing software system in terms of high-level models. Box and arrow sketches of a system, for instance, are often found on engineers' whiteboards. Although these models are commonly used, reasoning about the system in terms of such models can be dangerous because the models are almost always inaccurate with respect to the system's source.

Current reverse engineering systems derive high-level models from the source code. These derived models are useful because they are, by their very nature, accurate representations of the source. Although accurate, the models created by these reverse engineering systems may differ from the models sketched by engineers; an example of this is reported by Wong et al. [WTMS95].

We lack a **common vocabulary** to describe software architecture

Musée canadien de l'histoire
National Gallery of Canada
Notre-Dame Cathedral Basilica
HULL
BYWARD MARKET
Strathcona Park
CF Rideau Centre
University of Ottawa
SANDY HILL
Rue Laurier
Parliament Hill
Colonel By Dr
Ottawa
Ottawa City Hall
DOWNTOWN
GOLDEN TRIANGLE
Rideau River
Queen Elizabeth Dr
CENTRETOWN
LEBRETON FLATS
Canadian Museum of Nature
Springhurst Park
Springhurst Park
Albert St
CHINATOWN
Greyhound Canada
34
48
93
99
97
48
95
34
40
48
91
42
83
36
99
66
417
77
79
36


NICHE
ARMOIRE
TUB FILLER
TOWEL RING

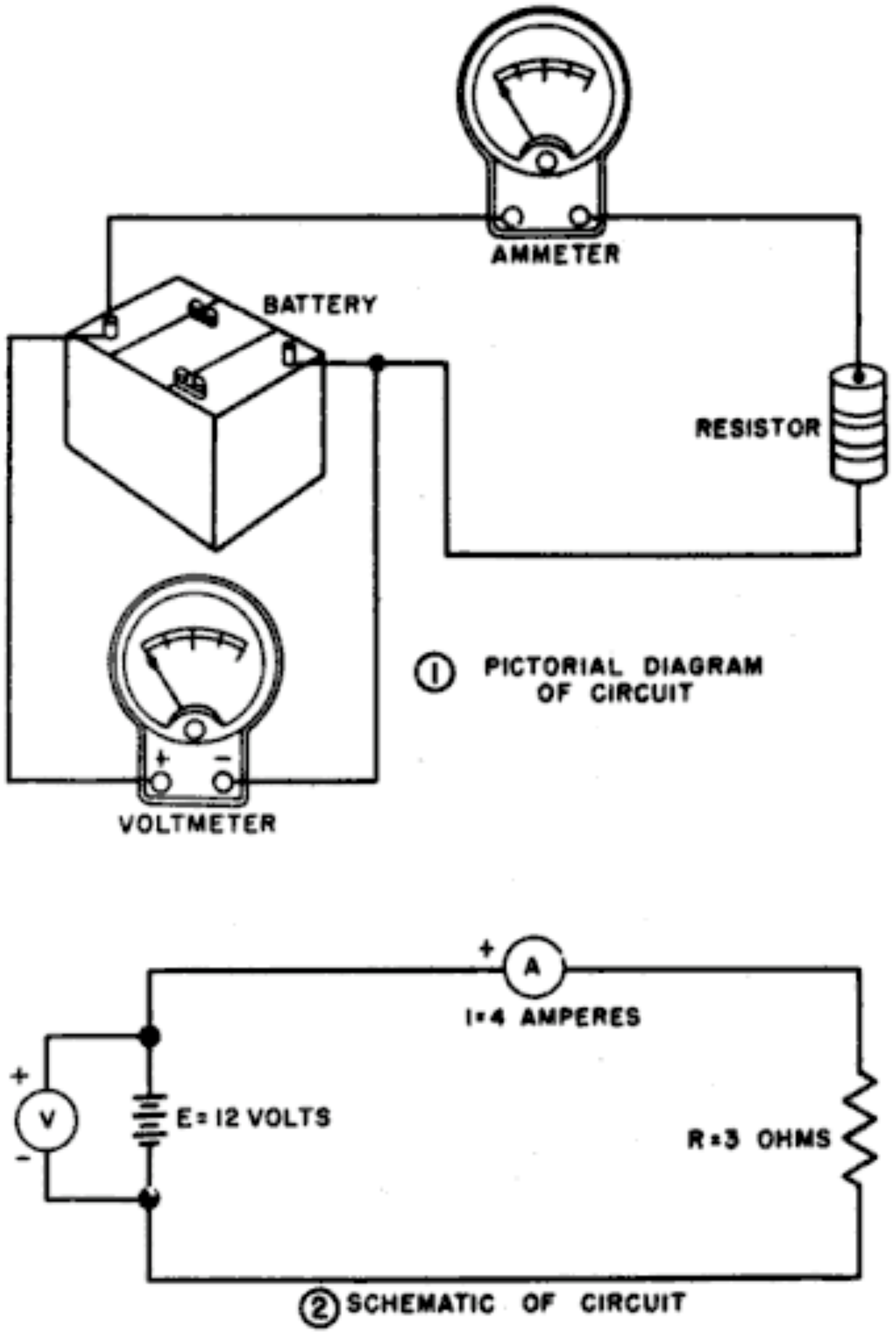
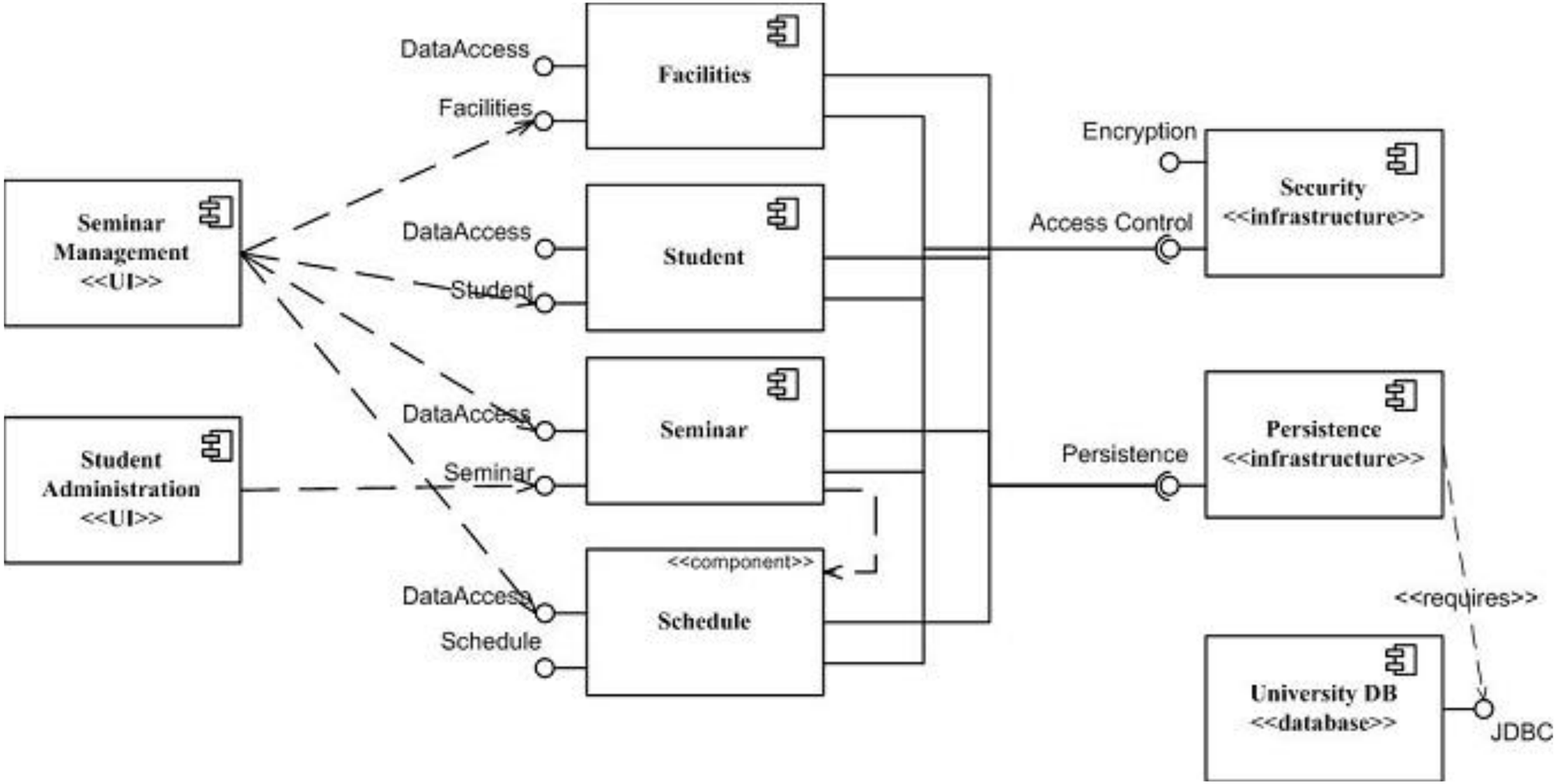
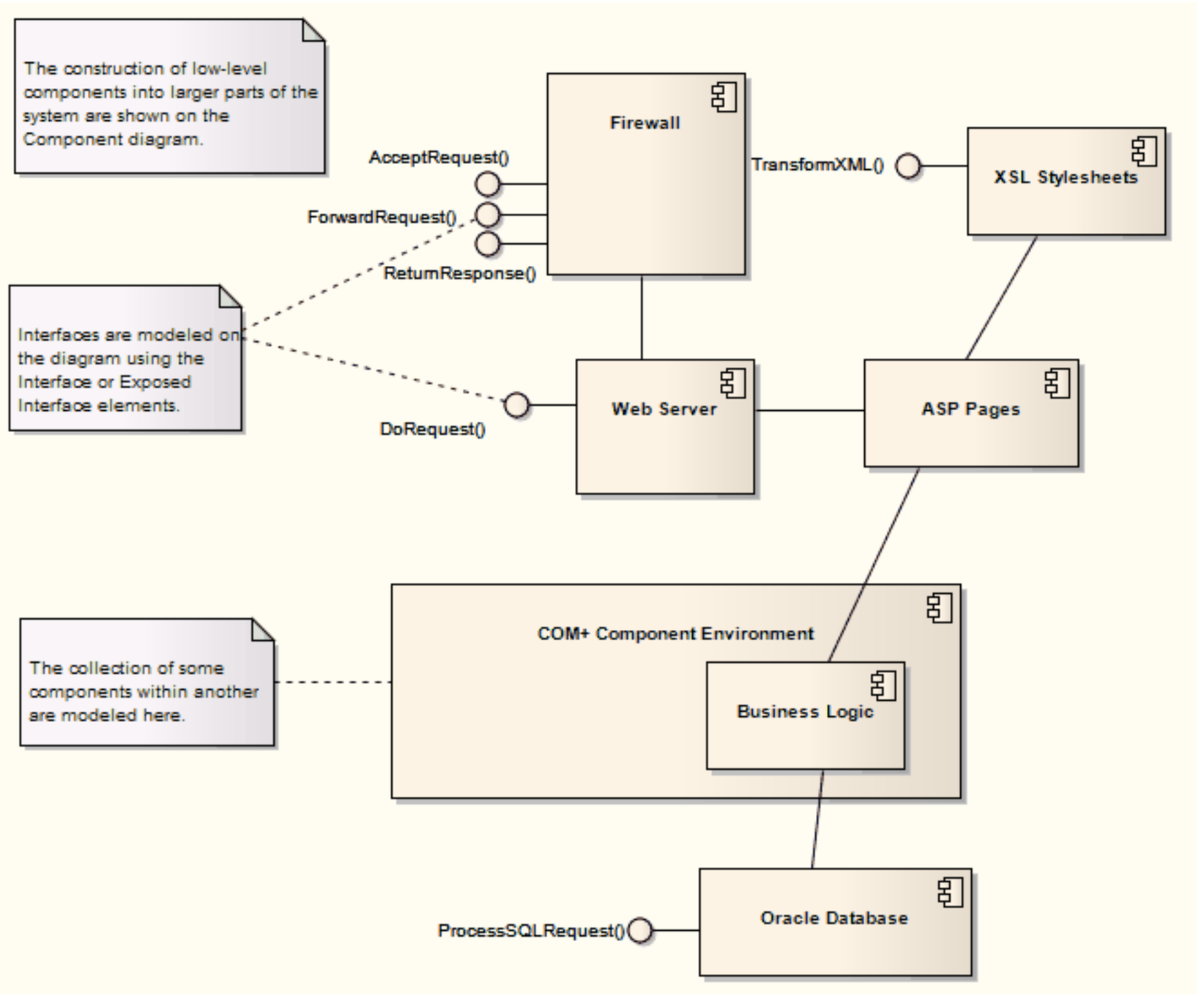
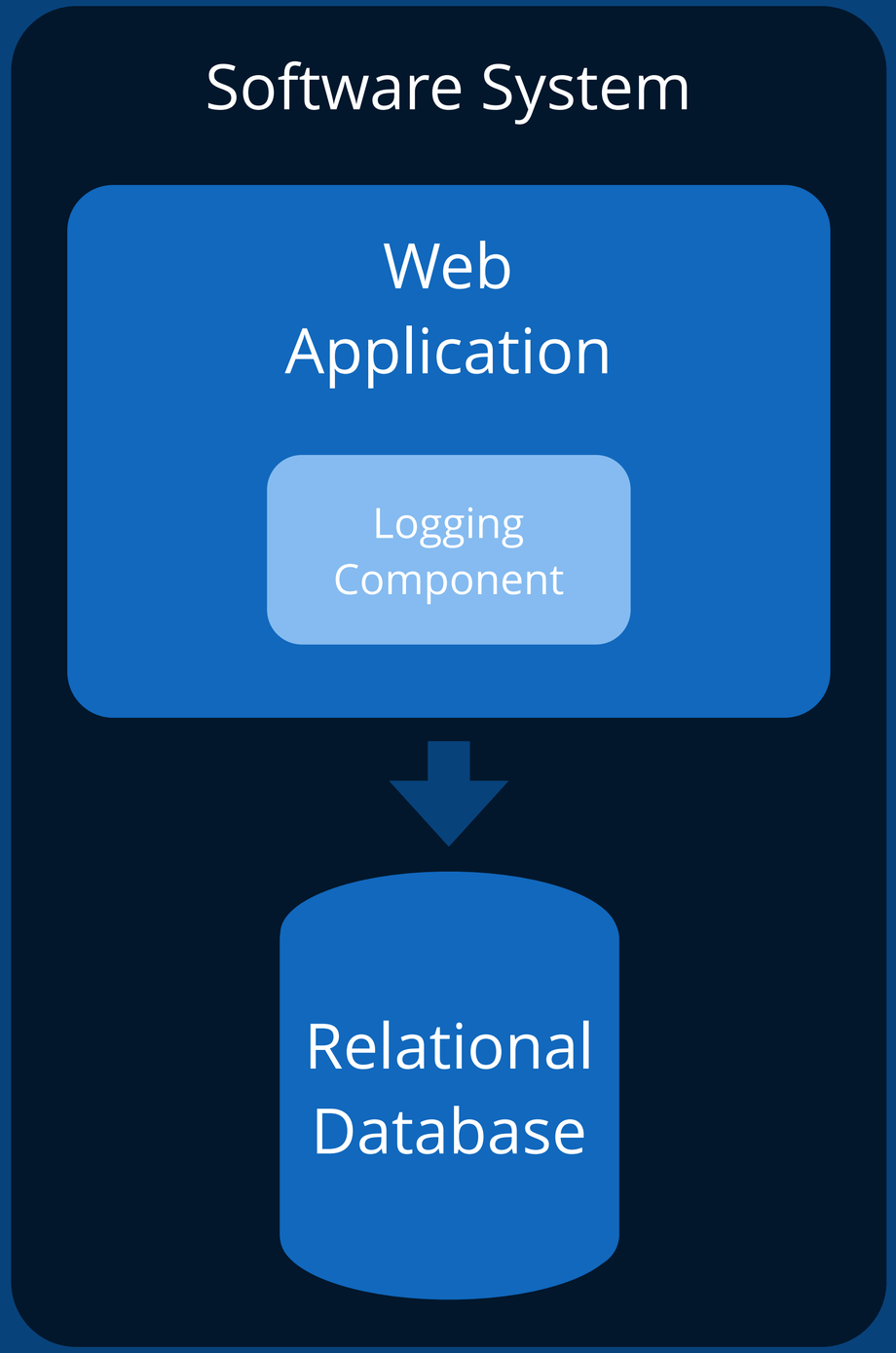



*Figure 48. Diagram of a basic circuit.*


DataAccess
Facilities
Facilities
Seminar Management <<UI>>
DataAccess
Student
Student
Student Administration <<UI>>
DataAccess
Seminar
Seminar
DataAccess
Schedule
Schedule
<<component>>
Encryption
Access Control
Security <<infrastructure>>
Persistence
Persistence <<infrastructure>>
<<requires>>
University DB <<database>>
JDBC


The construction of low-level components into larger parts of the system are shown on the Component diagram.
Firewall
AcceptRequest()
ForwardRequest()
ReturnResponse()
TransformXML()
XSL Stylesheets
Interfaces are modeled on the diagram using the Interface or Exposed Interface elements.
DoRequest()
Web Server
ASP Pages
The collection of some components within another are modeled here.
COM+ Component Environment
Business Logic
ProcessSQLRequest()
Oracle Database




# 1 component

*noun* | com·po·nent | \kəm-ˈpō-nənt, ˈkäm-ˌ, käm-ˈ\

## Simple Definition of COMPONENT

Popularity: Top 30% of words

: one of the parts of something (such as a system or mixture) : an important piece of something

Source: Merriam-Webster's Learner's Dictionary

# Ubiquitous language

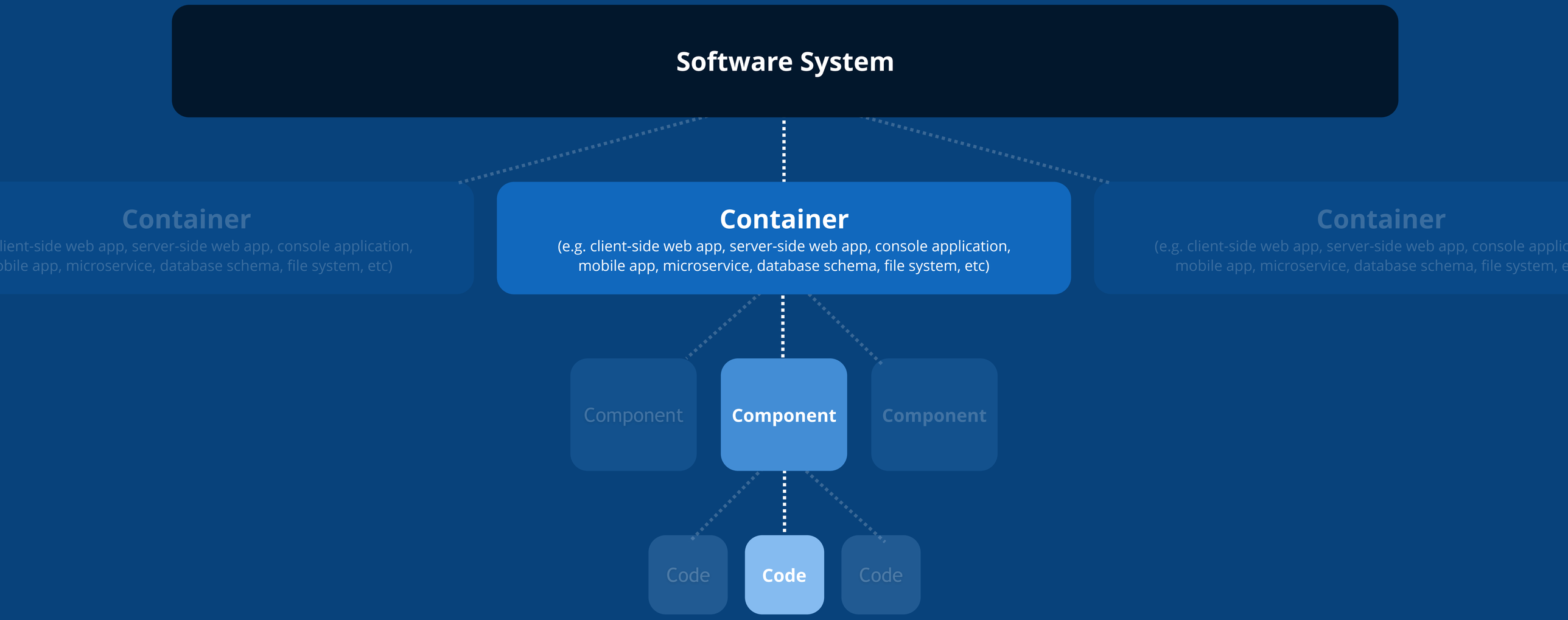
# Abstractions

# Would you code it that way?

(ensure that your diagrams reflect your implementation intent)

When drawing software architecture diagrams, think like a software developer

A **common set of abstractions**
is more important
than a common notation




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 **code elements**.

# Static structure diagrams

# C4

c4model.com

## 1. System Context

The system plus users and system dependencies.

## 2. Containers

The overall shape of the architecture and technology choices.

## 3. Components

Logical components and their interactions within a container.

## 4. Classes (or Code)

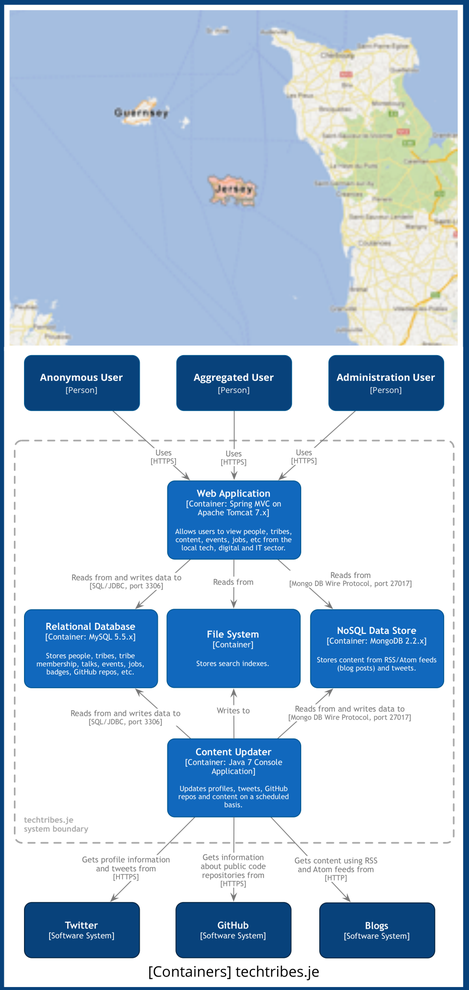
Component implementation details.

Overview first

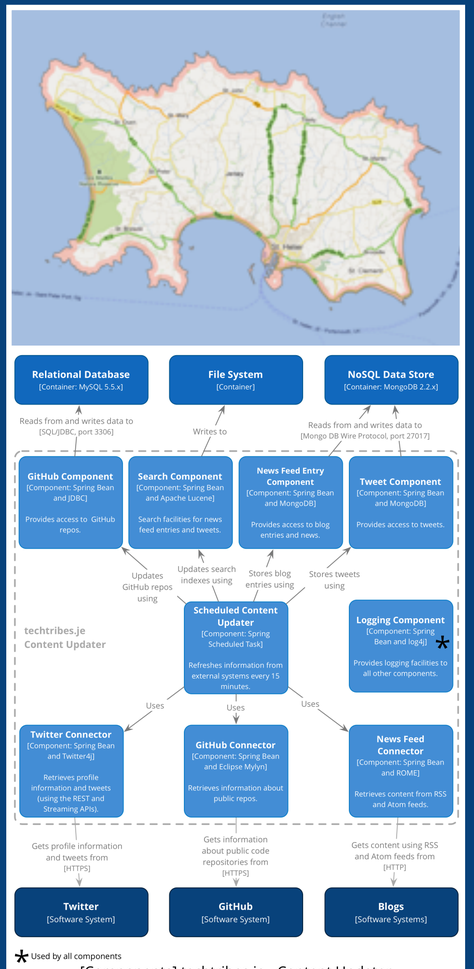
Zoom & filter

Details on demand

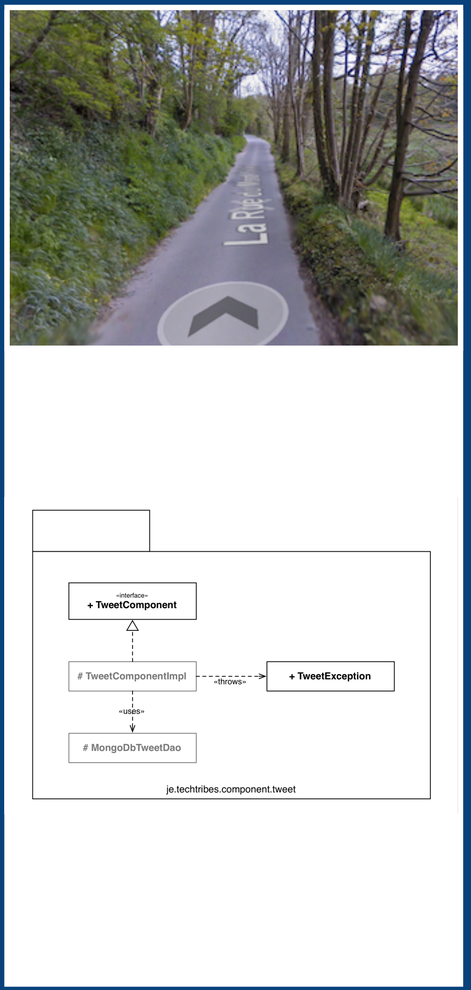
[System Context] techtribes.je



[Containers] techtribes.je

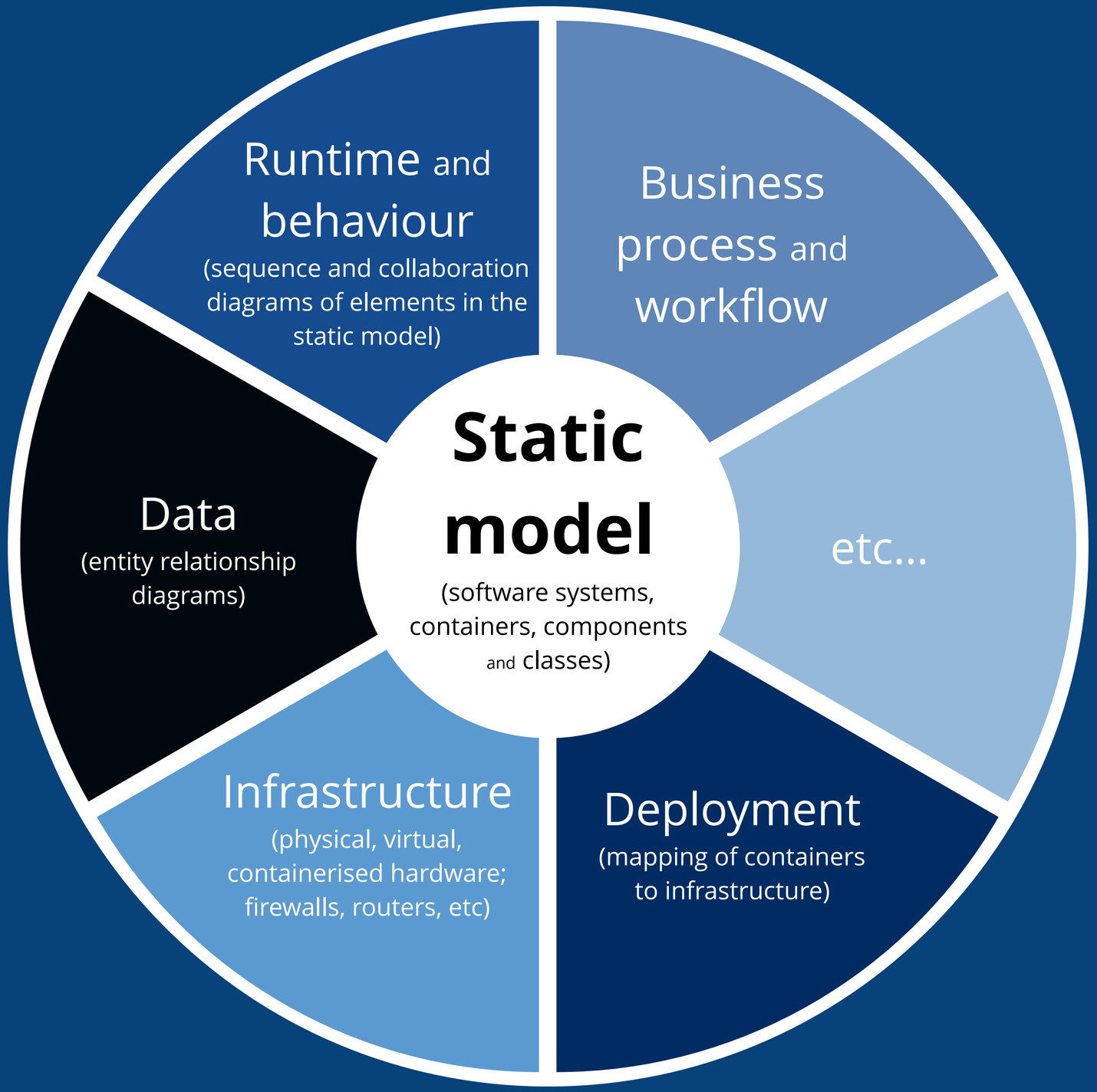


[Components] techtribes.je - Content Updater




# Diagrams are maps

that help software developers navigate a large and/or complex codebase



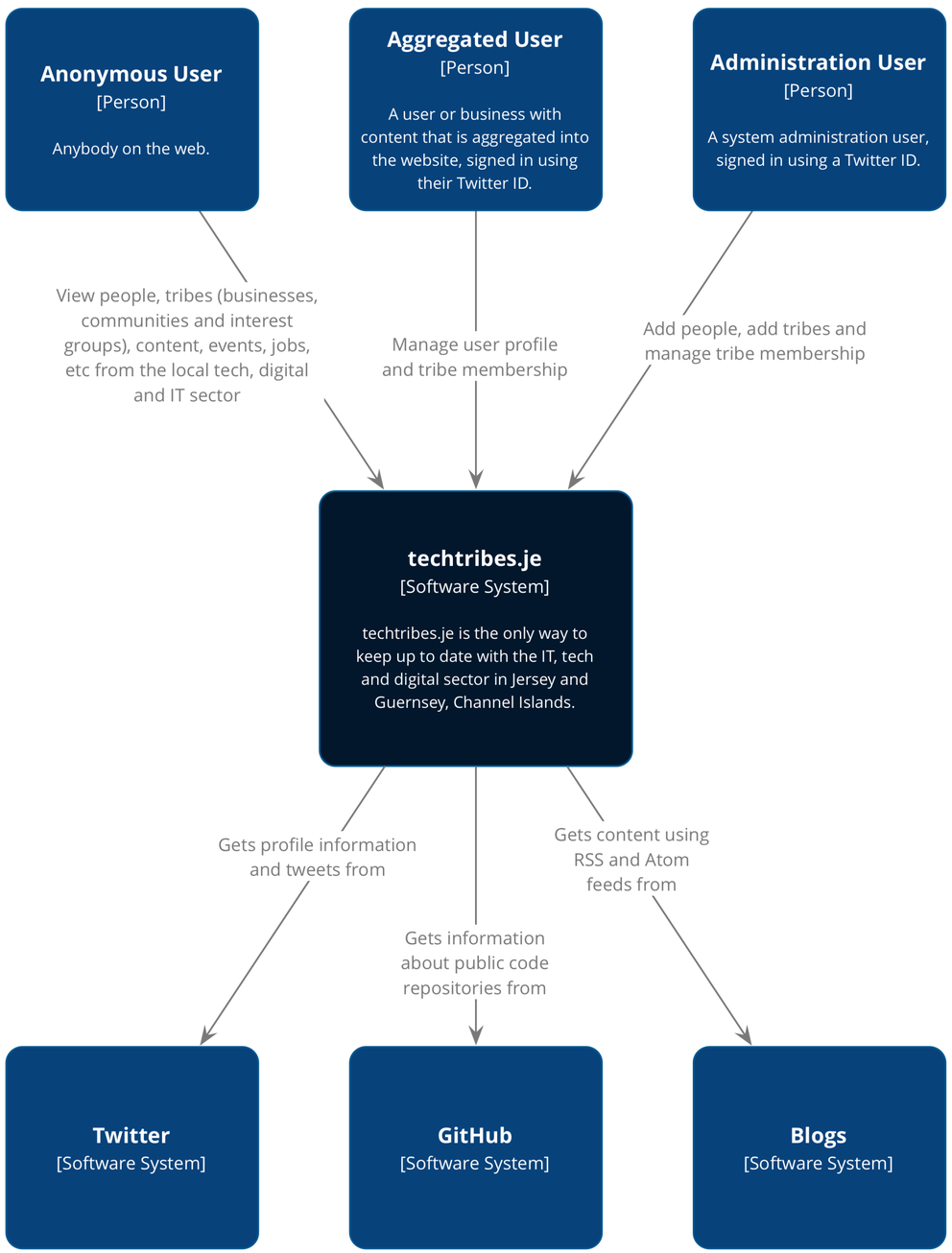

A model of the **static structure** forms the basis for other views

# Example

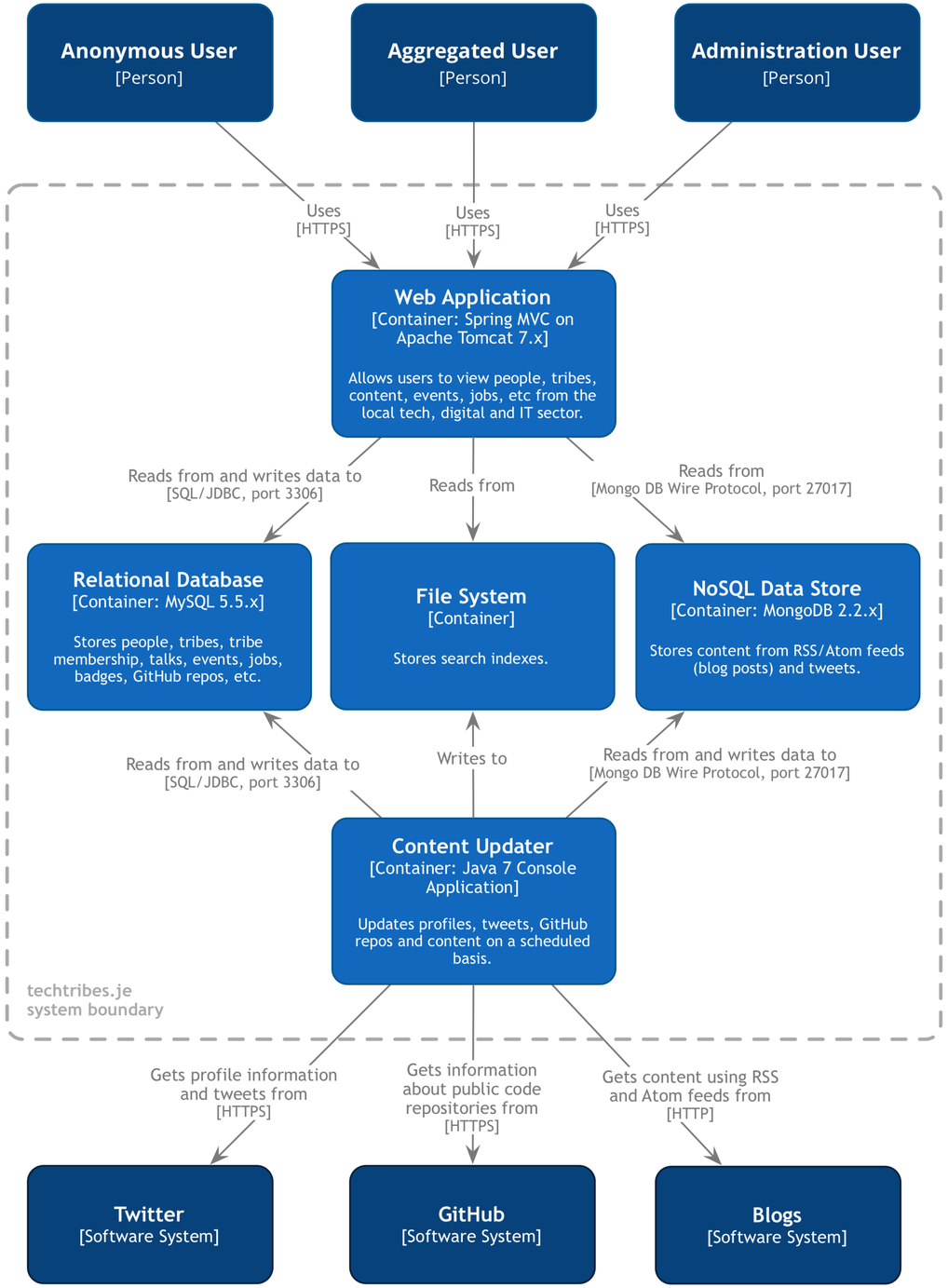
(techtribes.je)

# techtribes.je

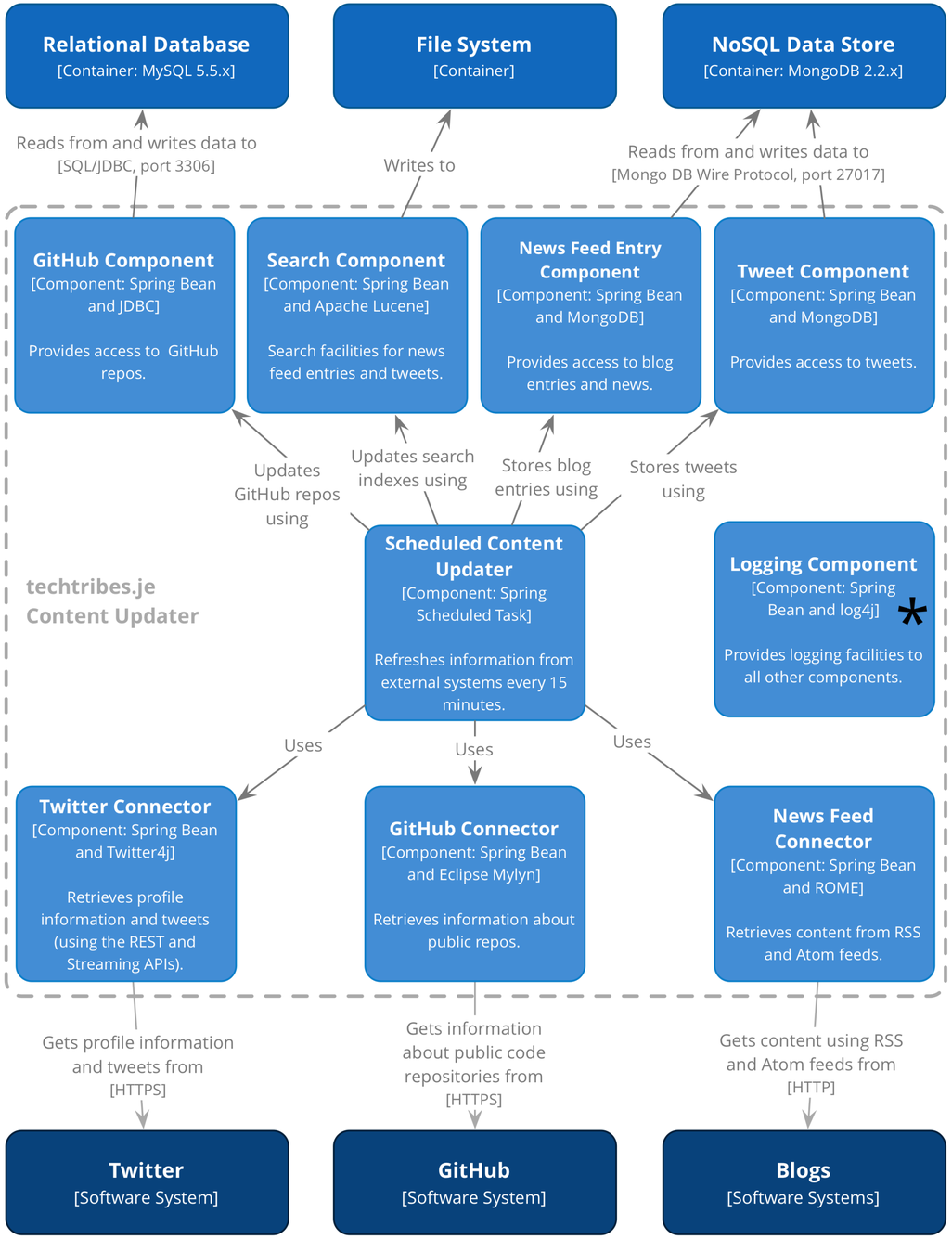
A simple content aggregator for the local tech and digital industry




[System Context] techtribes.je

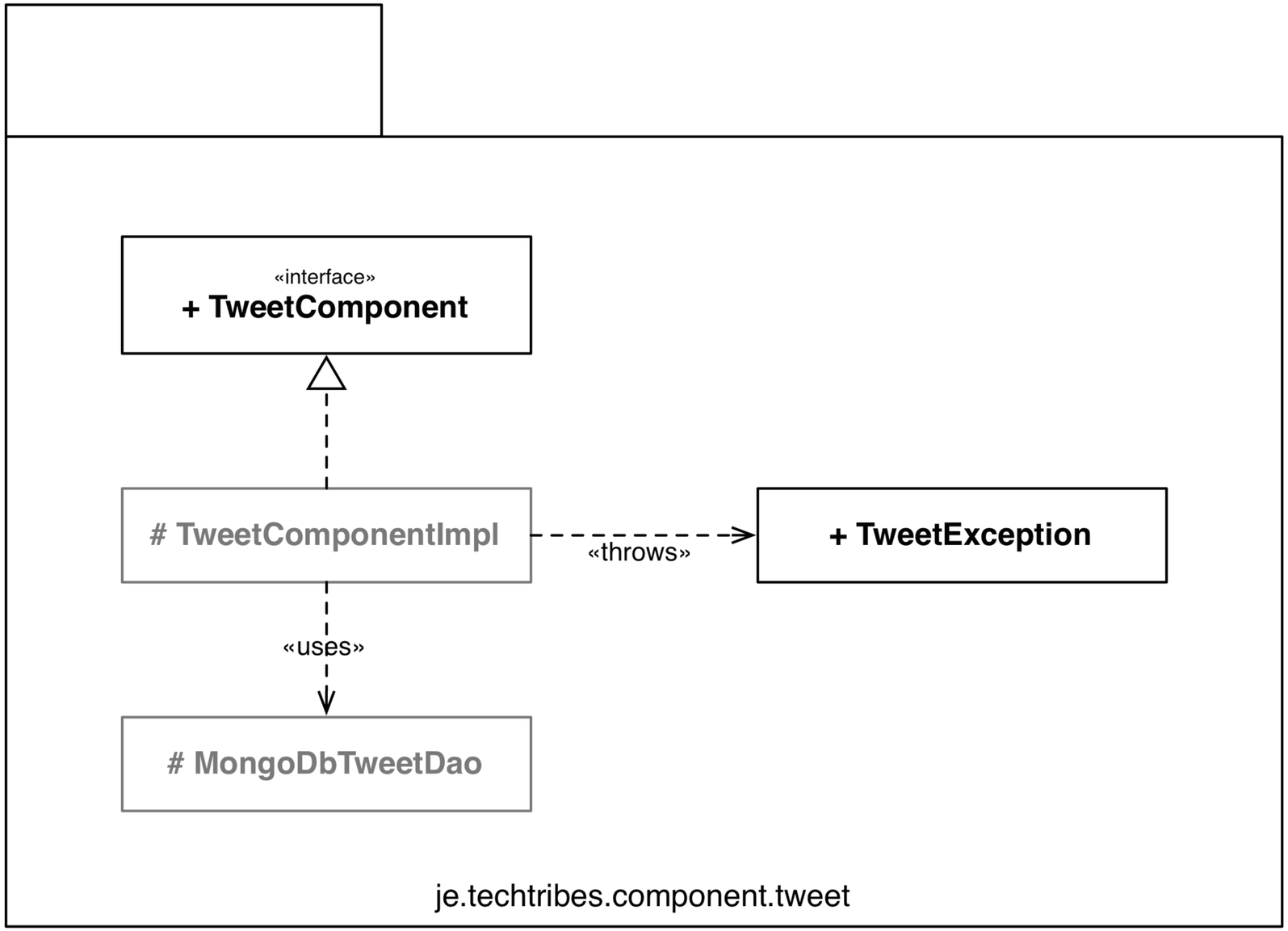



[Containers] techtribes.je




* Used by all components

[Components] techtribes.je - Content Updater


«interface»
+ TweetComponent
# TweetComponentImpl
«throws»
+ TweetException
«uses»
# MongoDbTweetDao
je.techtribes.component.tweet

# Notation

# Titles

Short and meaningful, include the **diagram type**, numbered if diagram order is important; for example:

**System Context diagram** for Financial Risk System

[**System Context**] Financial Risk System

# Layout

Sticky notes and index cards (e.g. CRC cards) make a great substitute for hand-drawn boxes, especially if you don’t have a whiteboard

# Orientation

Most important thing in the middle;
try to be consistent across diagrams

# Acronyms

Be wary of using acronyms, especially those related to the business/domain that you work in

# Elements

Start with simple boxes containing the element name, type, technology (if appropriate) and a description/responsibilities

## Anonymous User

[Person]

Anybody on the web.

## 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.

## Web Application

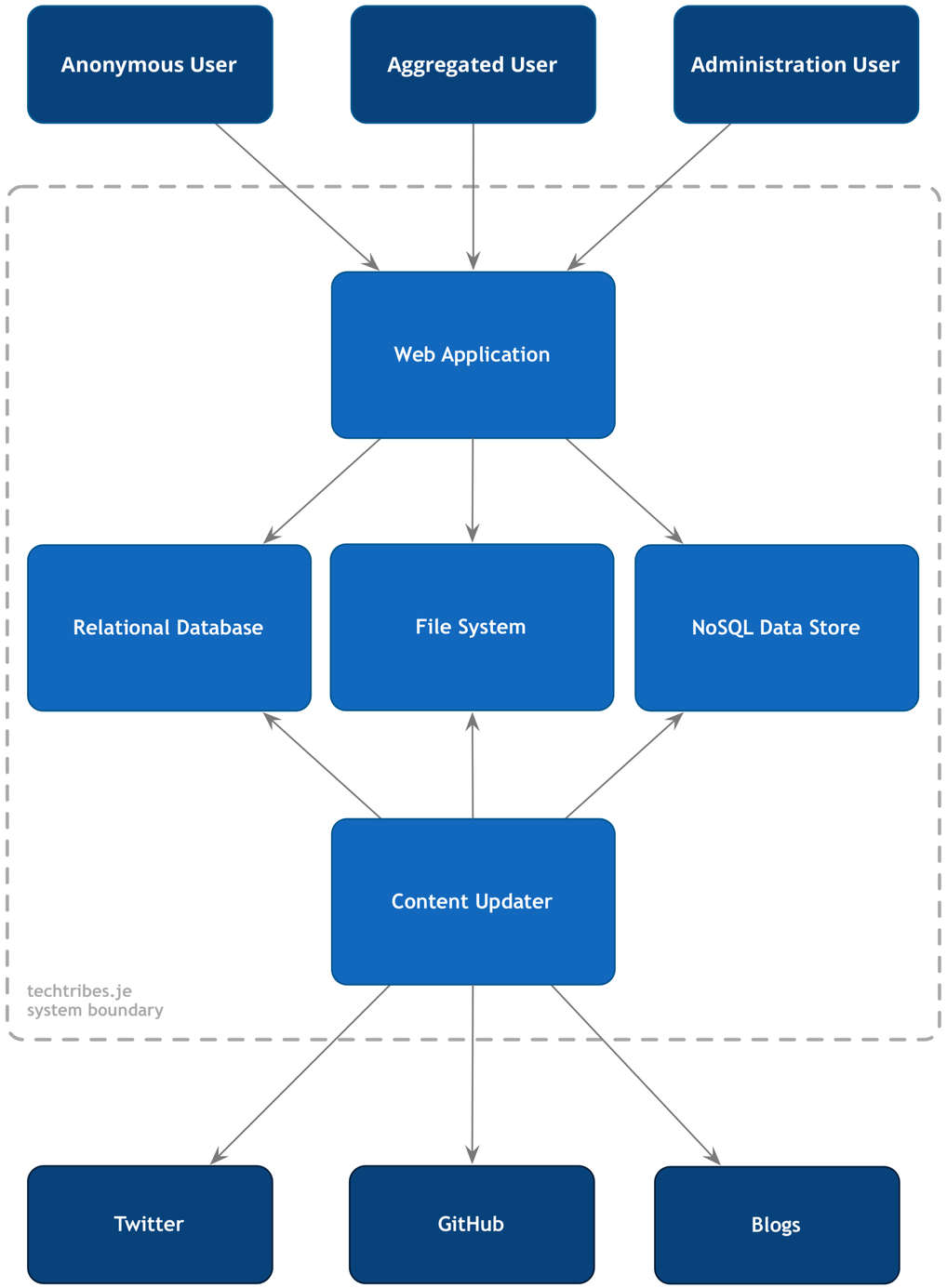
[Container: Java + Spring MVC]

Allows users to view people, tribes, content, events, jobs, etc from the local tech, digital and IT sector.

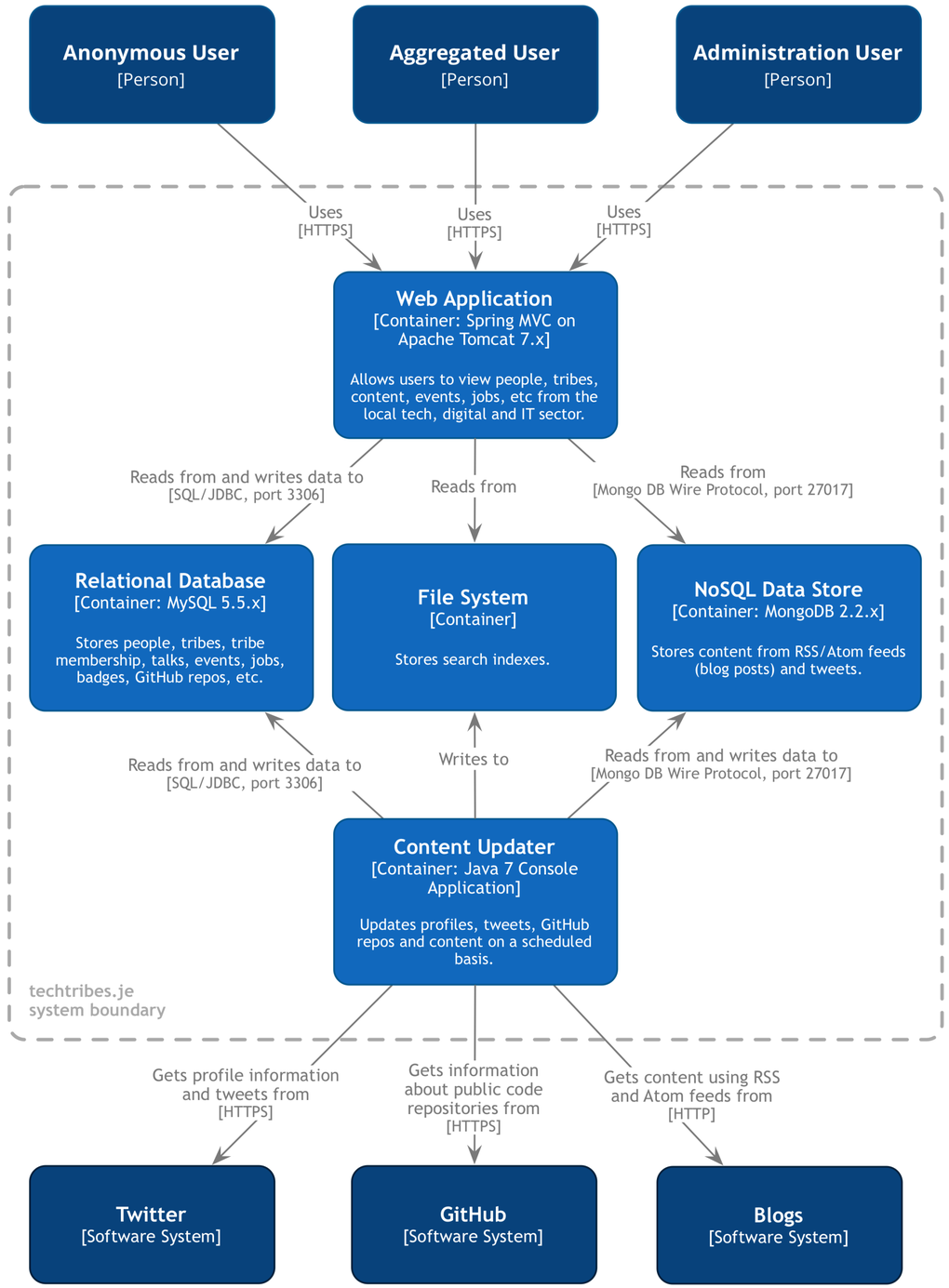
## Twitter Connector

[Component: Spring Bean + Twitter4j]

Retrieves profile information and tweets (using the REST and Streaming APIs).




[Containers] techtribes.je




[Containers] techtribes.je

# Lines

Favour uni-directional lines showing the most important dependencies or data flow, with an annotation to be explicit about the purpose of the line and direction

# Dependency vs data flow

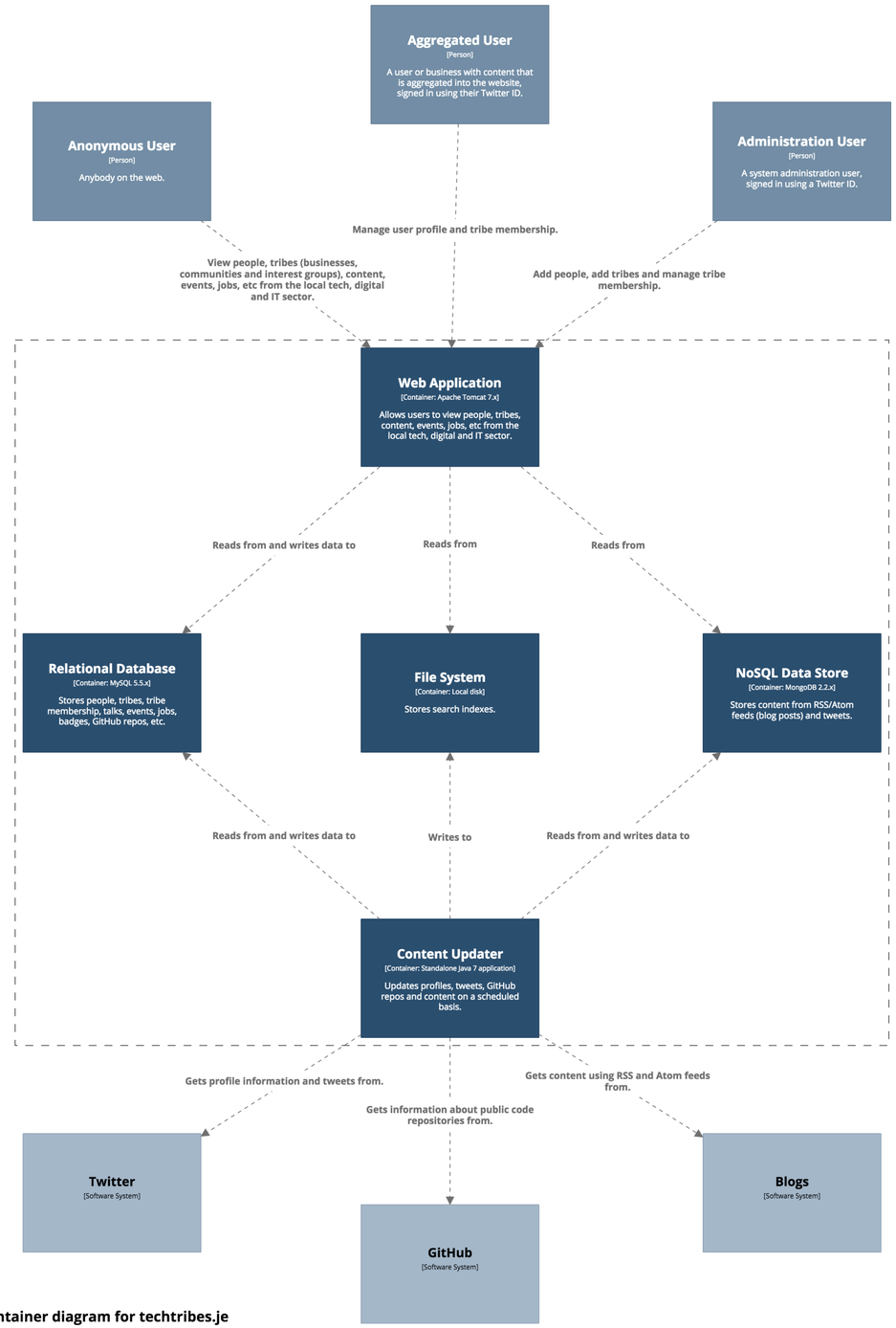
# Read the relationship out loud

# Key/legend

Explain shapes, line styles, colours, borders, acronyms, etc

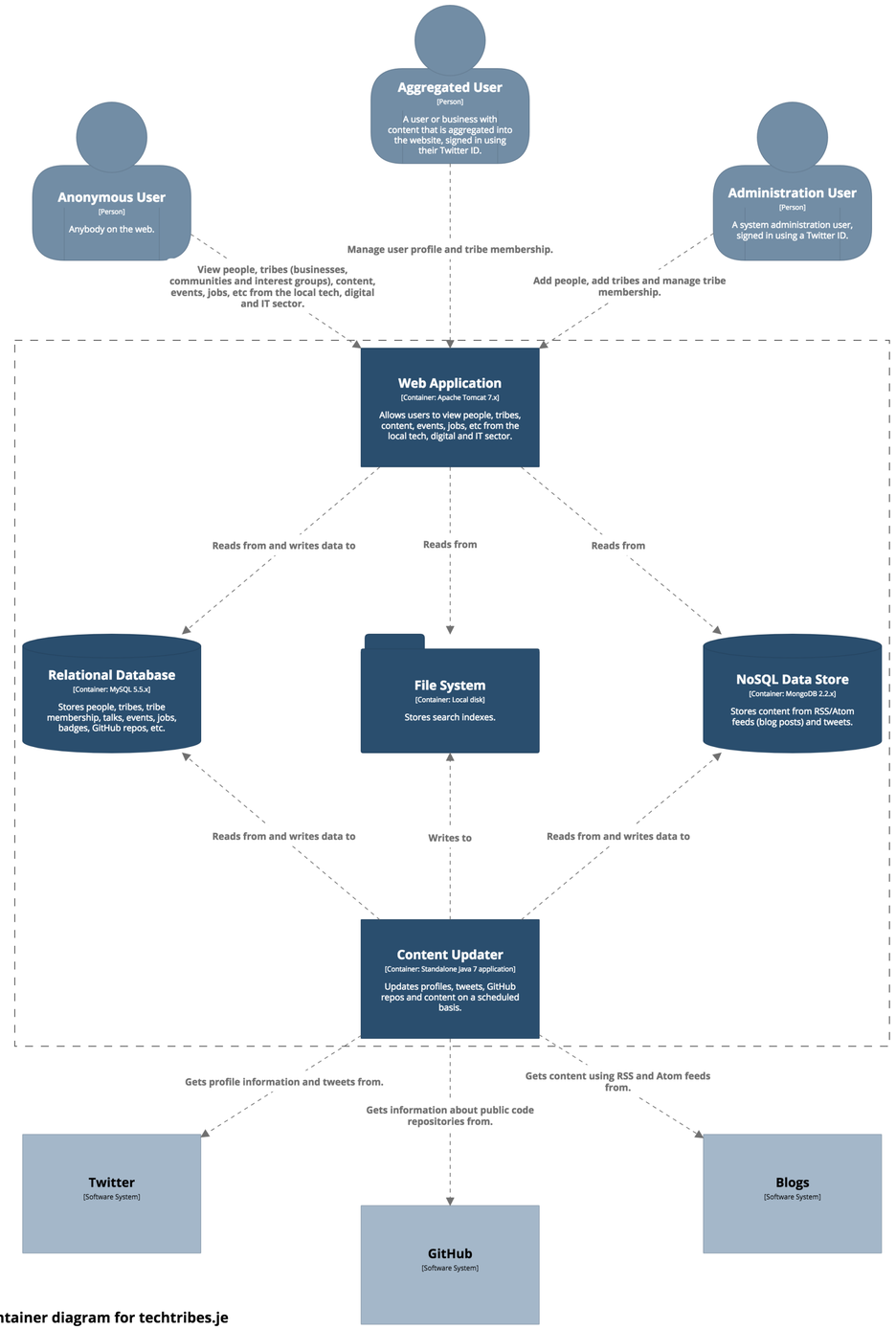
... even if your notation seems obvious!

Use shape, colour and size to **complement** a diagram that already makes sense




**Container diagram for techtribes.je**

Friday 12 May 2017 10:42 UTC




**Container diagram for techtribes.je**

Monday 27 February 2017 09:39 UTC

Increase the **readability** of software architecture diagrams, so they can **stand alone**

Any narrative should **complement** the diagram rather than explain it

# Draw a Component diagram to describe a container

(components within the container in scope)

**30 minutes**

Designing software is where the complexity should be, not communicating it!

Richer diagrams lead to richer **design discussions**

# The perfection game

We rate the diagrams... (1-10)

We liked...

To make the diagrams perfect...

**15 minutes**

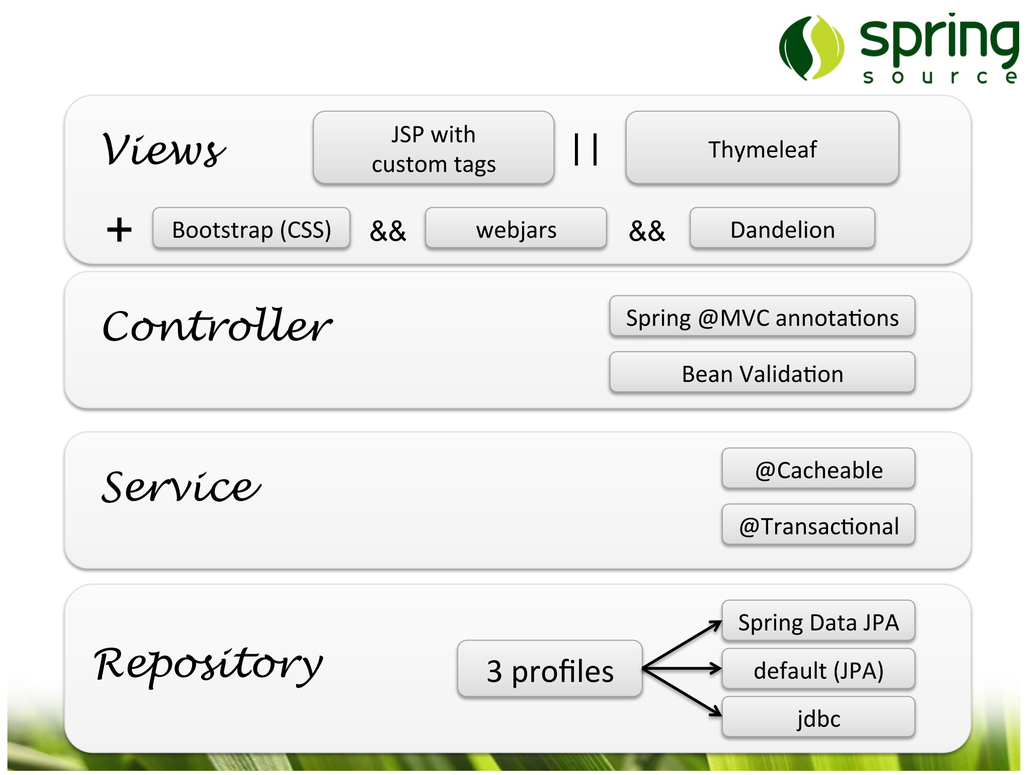
Similar levels of abstraction provide a way to easily **compare** solutions

# C4 and UML

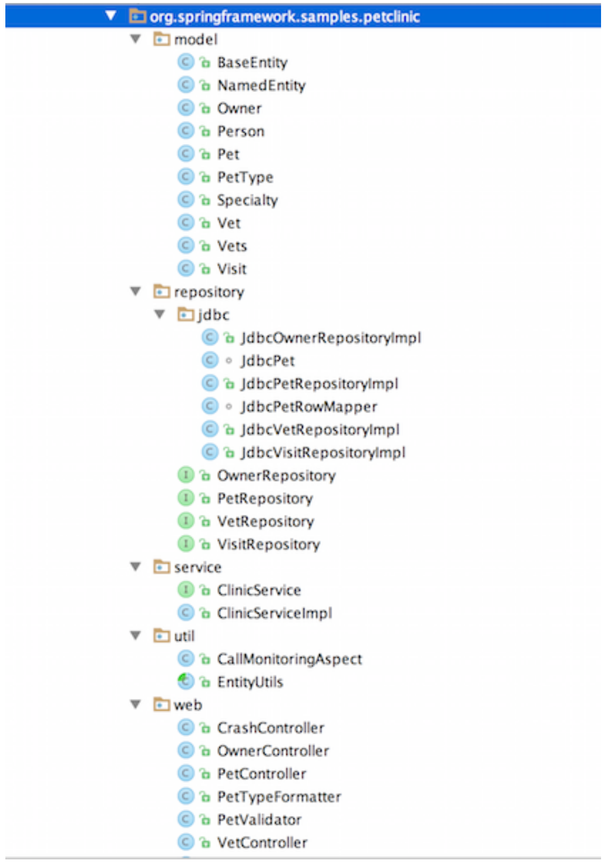
# Spring PetClinic

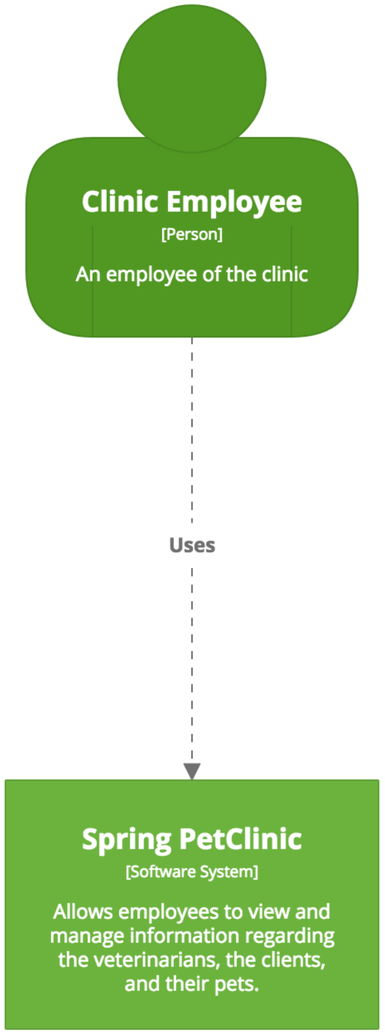
A sample application that illustrates how to build Java web applications using the Spring MVC framework

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



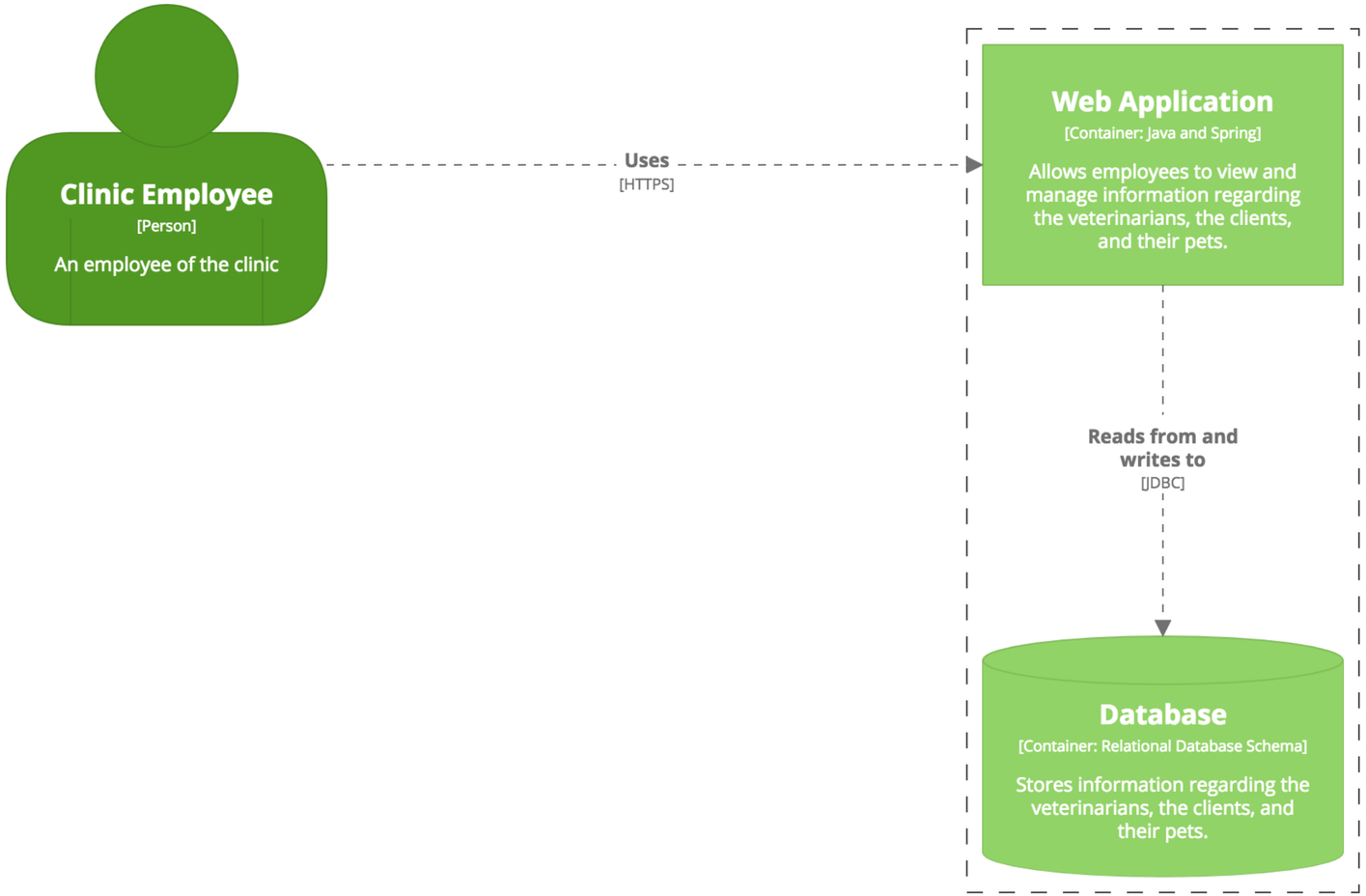

https://speakerdeck.com/michaelisvy/spring-petclinic-sample-application





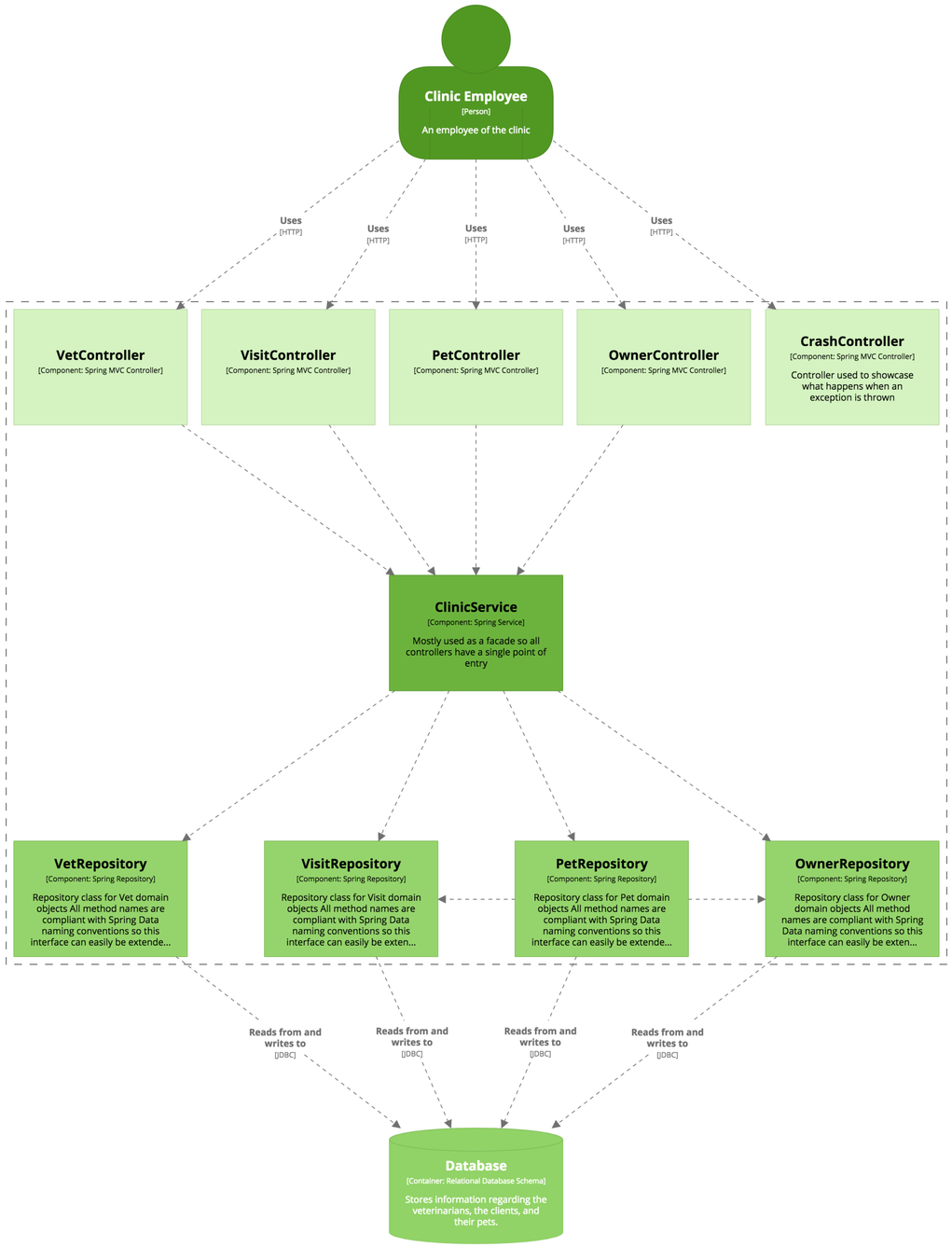

## System Context diagram for Spring PetClinic

The System Context diagram for the Spring PetClinic system.
Last modified: Monday 26 June 2017 12:59 UTC | Version: 95de1d9f8bf63560915331664b27a4a75ce1f1f6




## Container diagram for Spring PetClinic

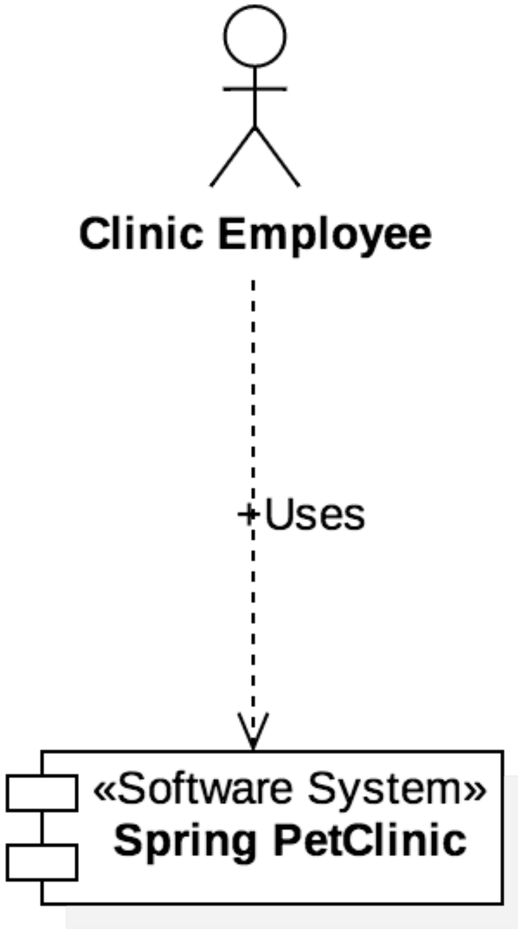
The Containers diagram for the Spring PetClinic system.
Last modified: Monday 26 June 2017 12:59 UTC | Version: 95de1d9f8bf63560915331664b27a4a75ce1f1f6

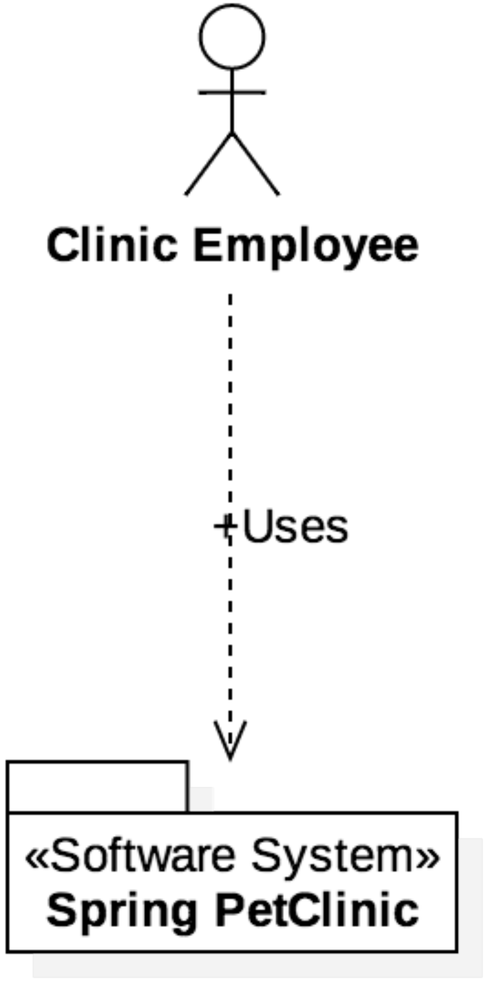


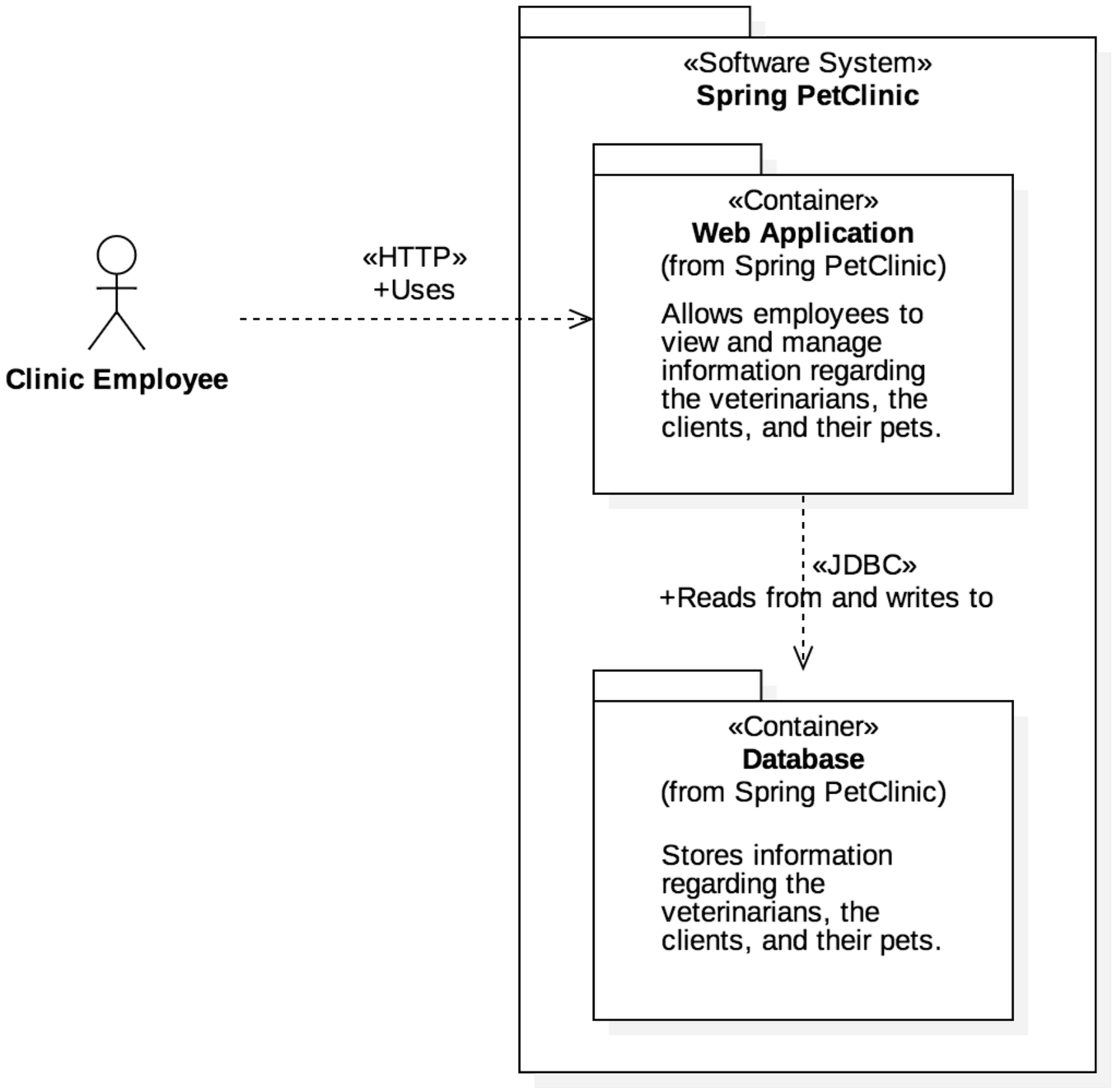

## Component diagram for Spring PetClinic - Web Application

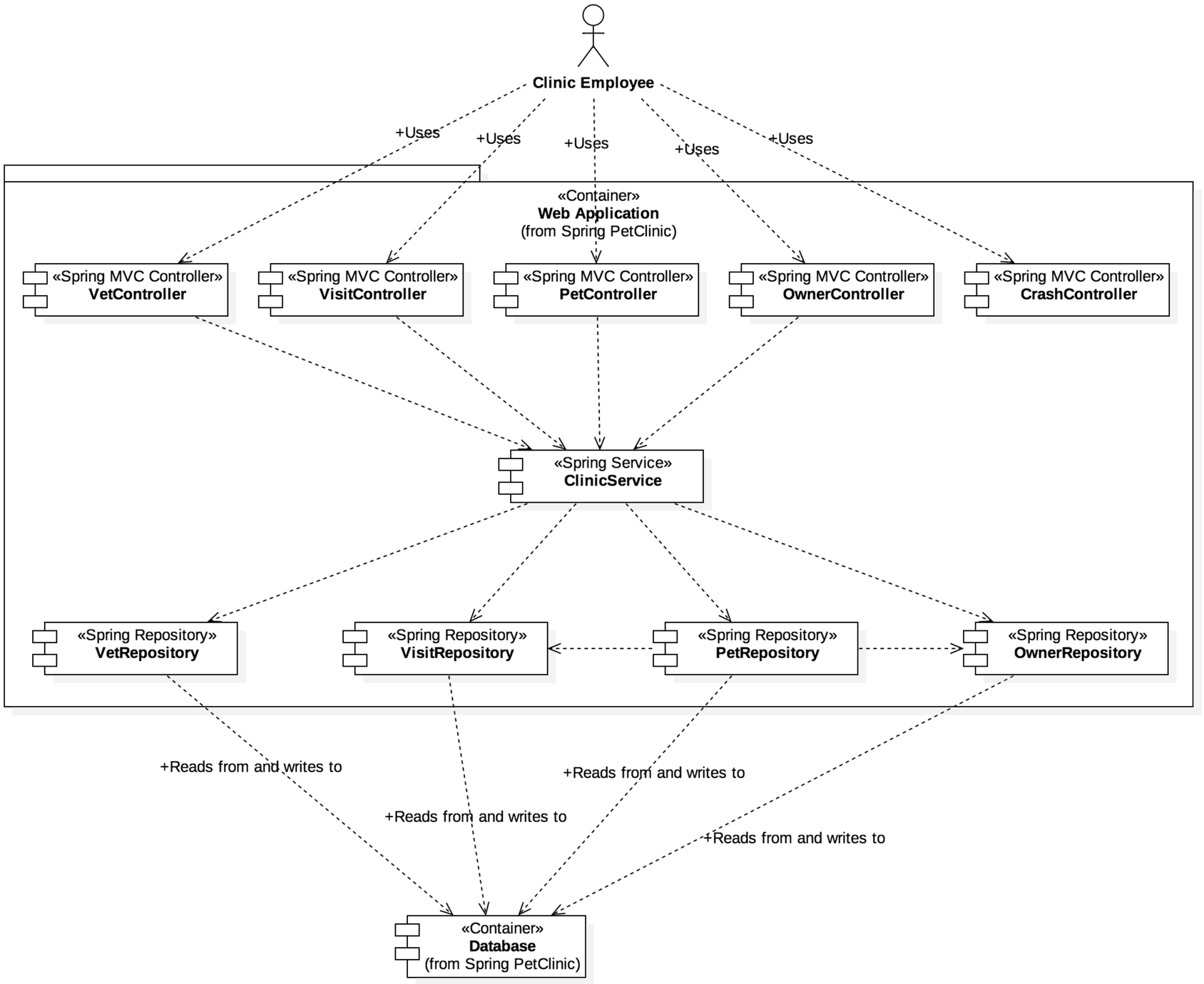
The Components diagram for the Spring PetClinic web application.

Last modified: Monday 26 June 2017 12:59 UTC | Version: 95de1d9f8bf63560915331664b27a4a75ce1f1f6

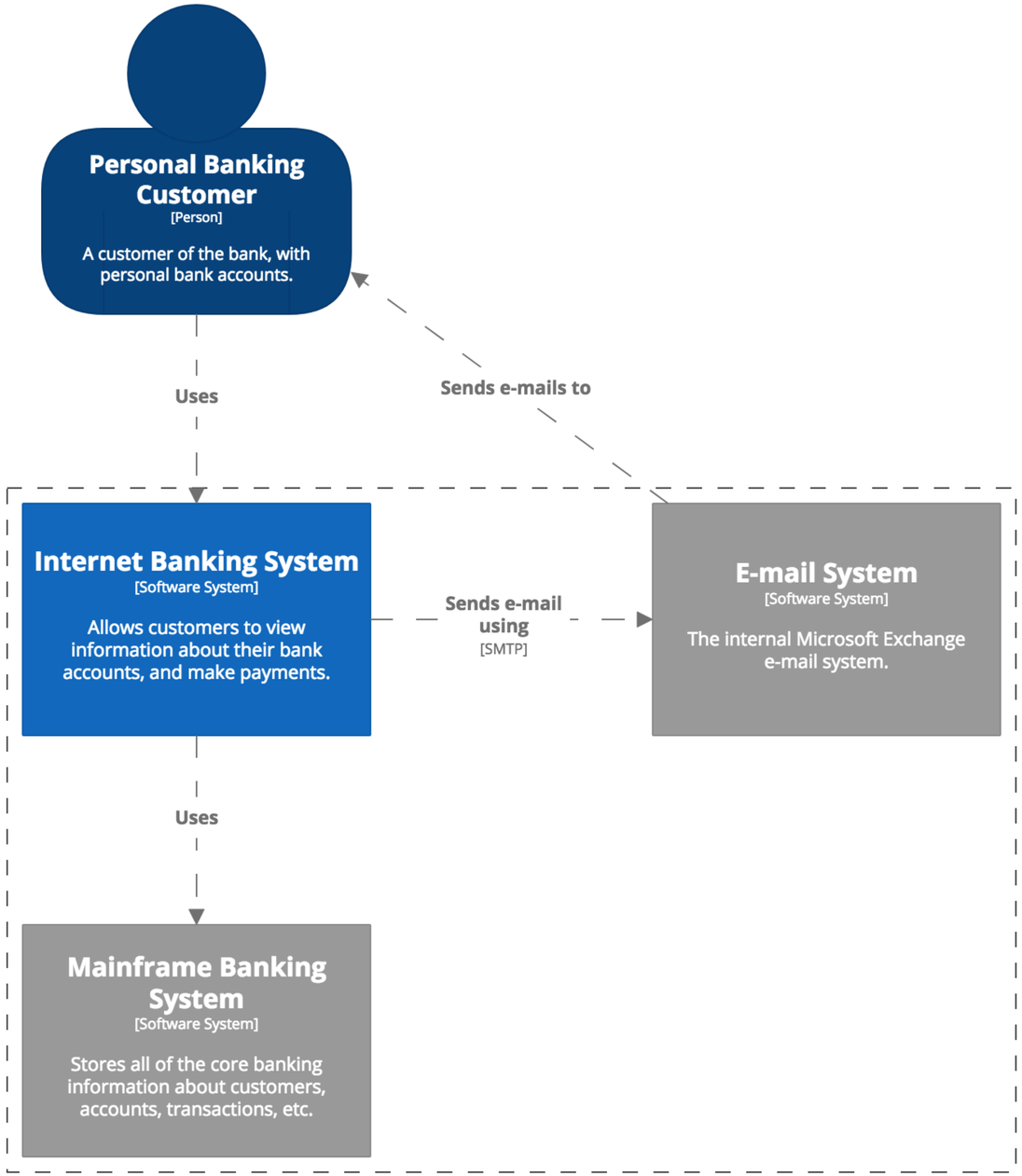

Clinic Employee
+Uses
«Software System»
Spring PetClinic


Clinic Employee
+Uses
«Software System»
Spring PetClinic


«Software System»
Spring PetClinic
«Container»
Web Application
(from Spring PetClinic)
Allows employees to view and manage information regarding the veterinarians, the clients, and their pets.
«HTTP»
+Uses
Clinic Employee
«JDBC»
+Reads from and writes to
«Container»
Database
(from Spring PetClinic)
Stores information regarding the veterinarians, the clients, and their pets.

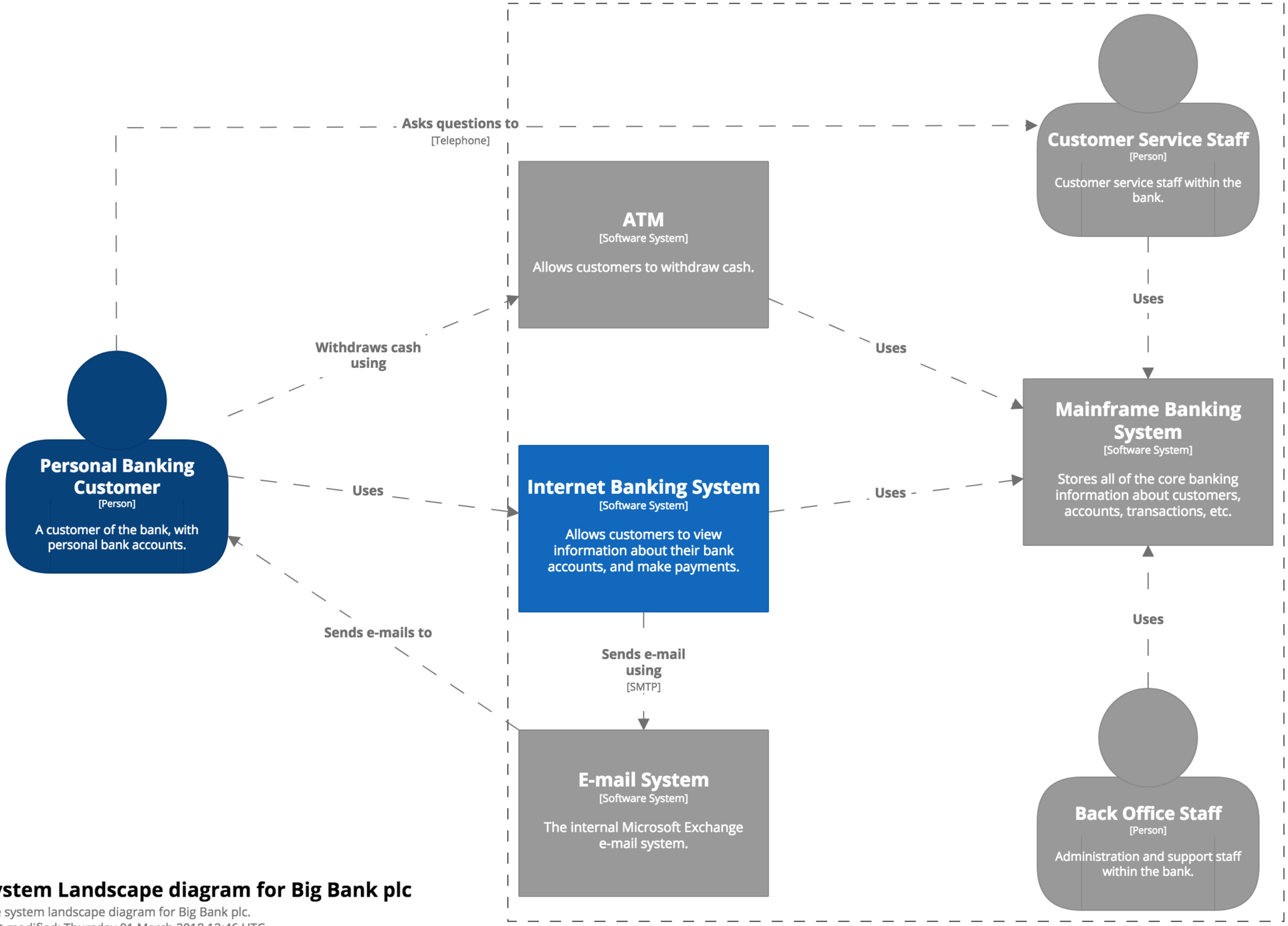

Clinic Employee
+Uses
+Uses
+Uses
+Uses
+Uses
«Container»
Web Application
(from Spring PetClinic)
«Spring MVC Controller»
VetController
«Spring MVC Controller»
VisitController
«Spring MVC Controller»
PetController
«Spring MVC Controller»
OwnerController
«Spring MVC Controller»
CrashController
«Spring Service»
ClinicService
«Spring Repository»
VetRepository
«Spring Repository»
VisitRepository
«Spring Repository»
PetRepository
«Spring Repository»
OwnerRepository
+Reads from and writes to
+Reads from and writes to
+Reads from and writes to
+Reads from and writes to
«Container»
Database
(from Spring PetClinic)

# System landscape diagrams




## System Context diagram for Internet Banking System

The system context diagram for the Internet Banking System.
Last modified: Thursday 01 March 2018 12:46 UTC

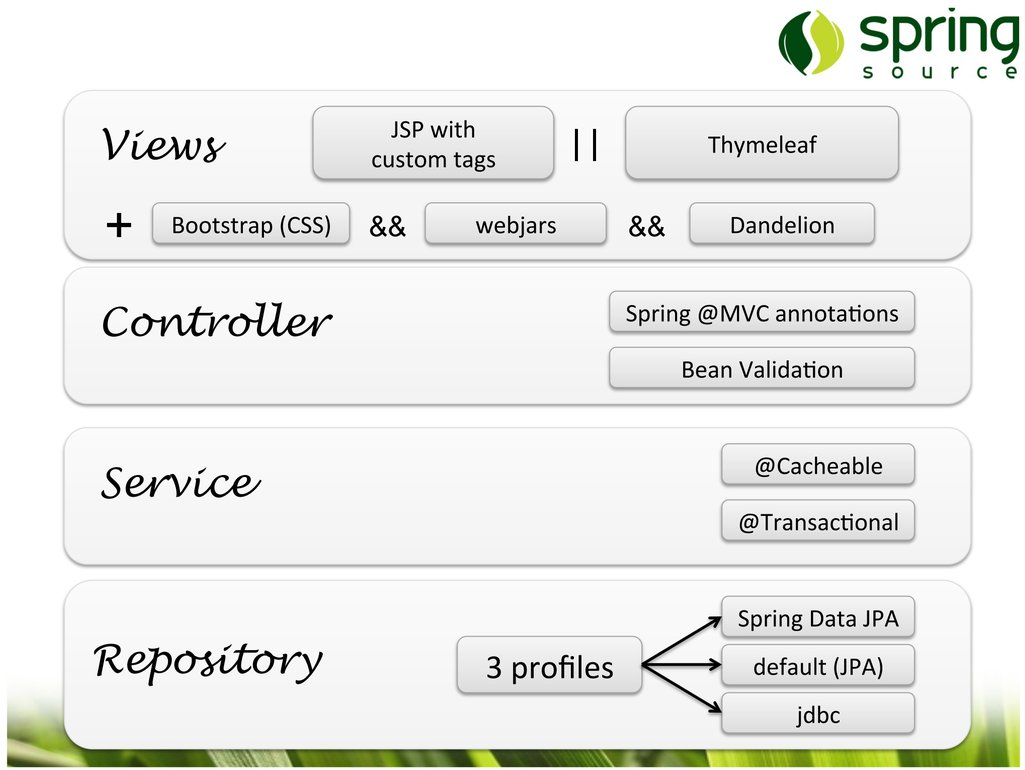


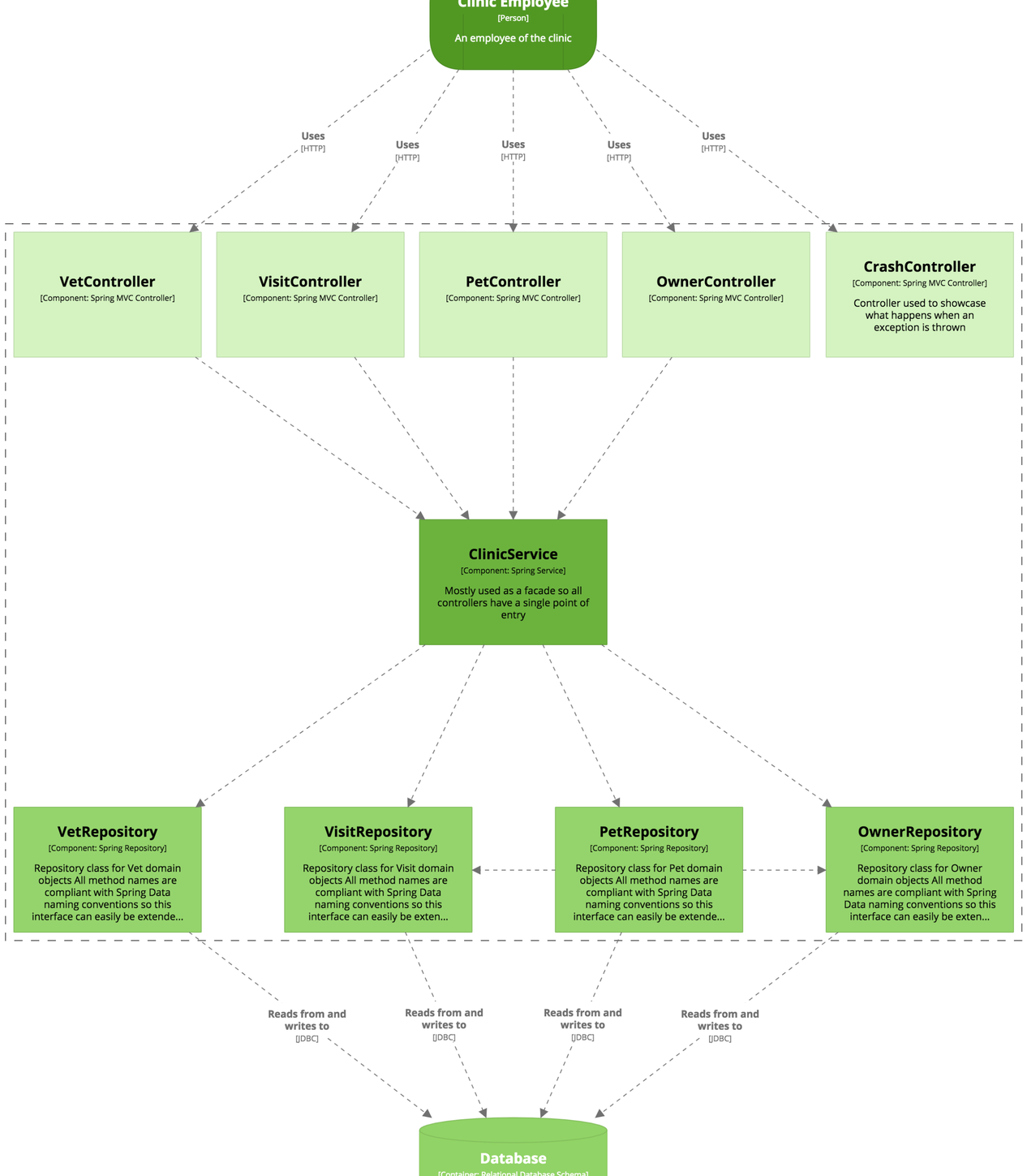

## System Landscape diagram for Big Bank plc

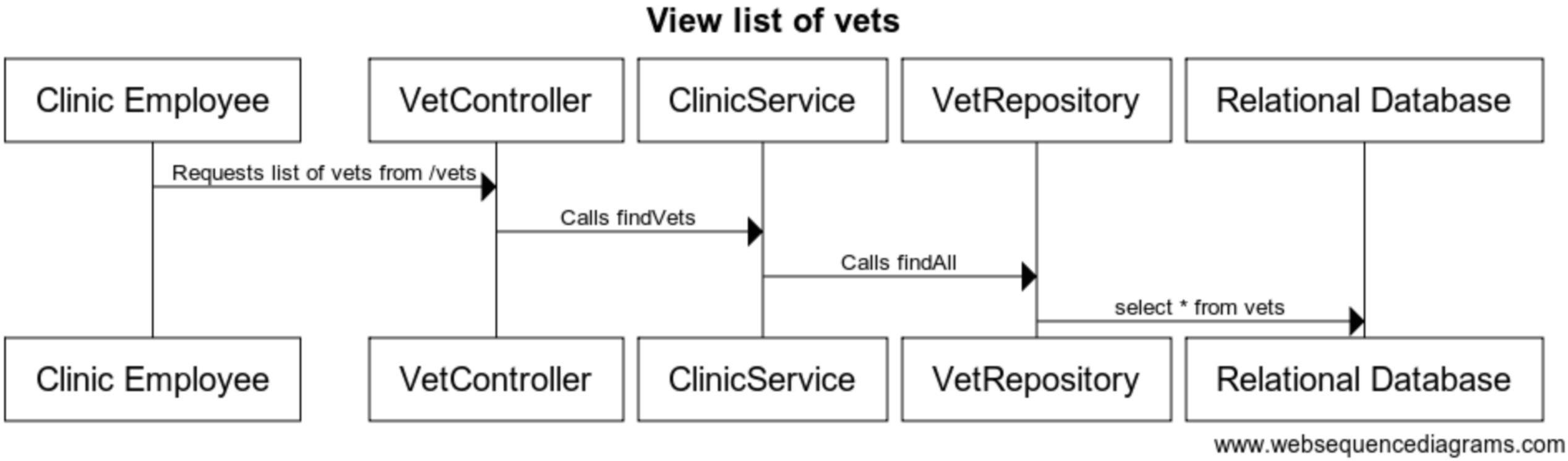
The system landscape diagram for Big Bank plc.
Last modified: Thursday 01 March 2018 12:46 UTC

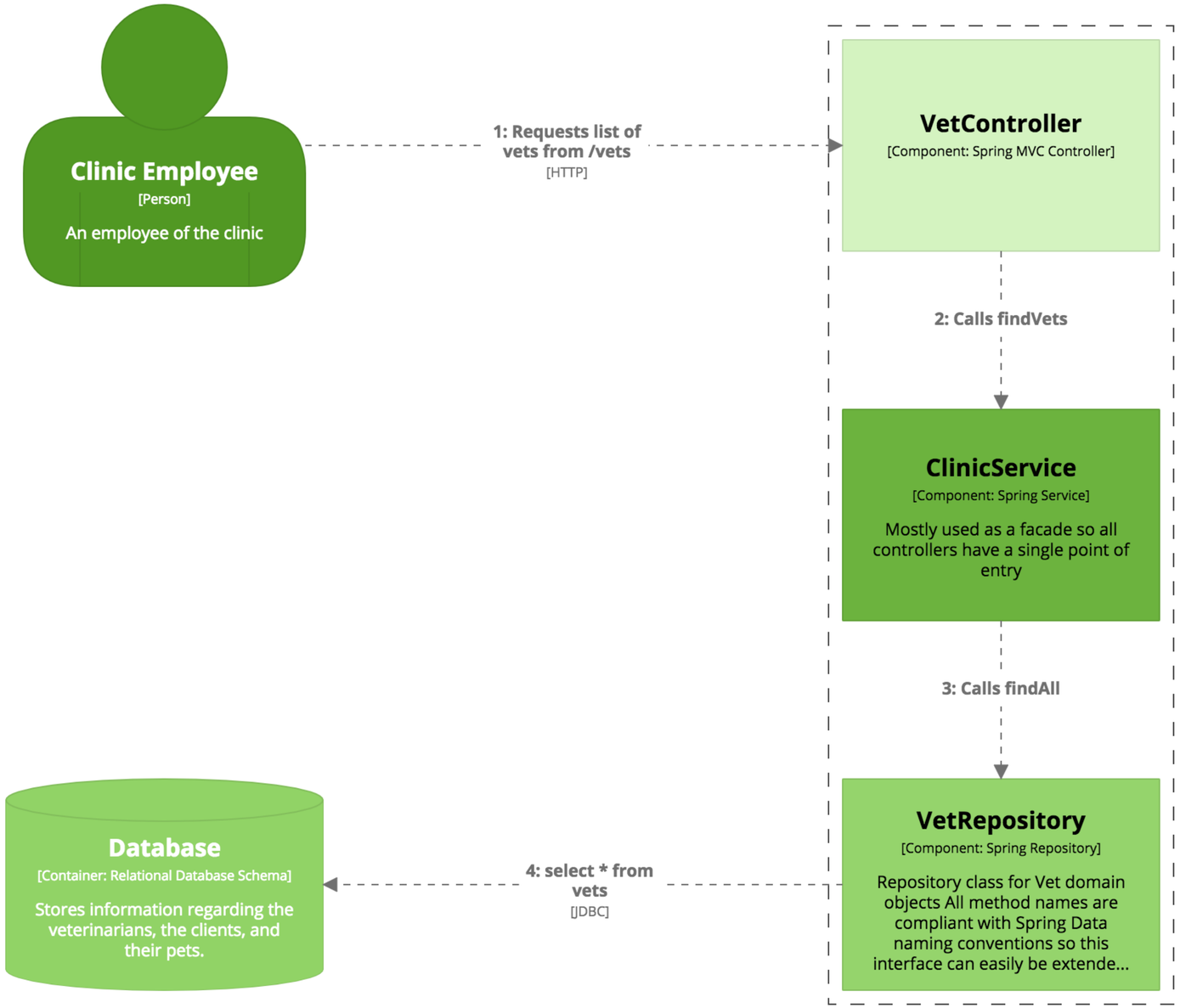
# Runtime/behavioural diagrams

Static structure diagrams are very useful, but they don’t tell the whole story


spring source
Views
JSP with custom tags
||
Thymeleaf
+
Bootstrap (CSS)
&&
webjars
&&
Dandelion
Controller
Spring @MVC annotations
Bean Validation
Service
@Cacheable
@Transactional
Repository
3 profiles
Spring Data JPA
default (JPA)
jdbc


Clinic Employee
[Person]
An employee of the clinic
Uses
[HTTP]
VetController
[Component: Spring MVC Controller]
VisitController
[Component: Spring MVC Controller]
PetController
[Component: Spring MVC Controller]
OwnerController
[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
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 extende...
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 exten...
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 extende...
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 exten...
Reads from and writes to
[JDBC]
Database

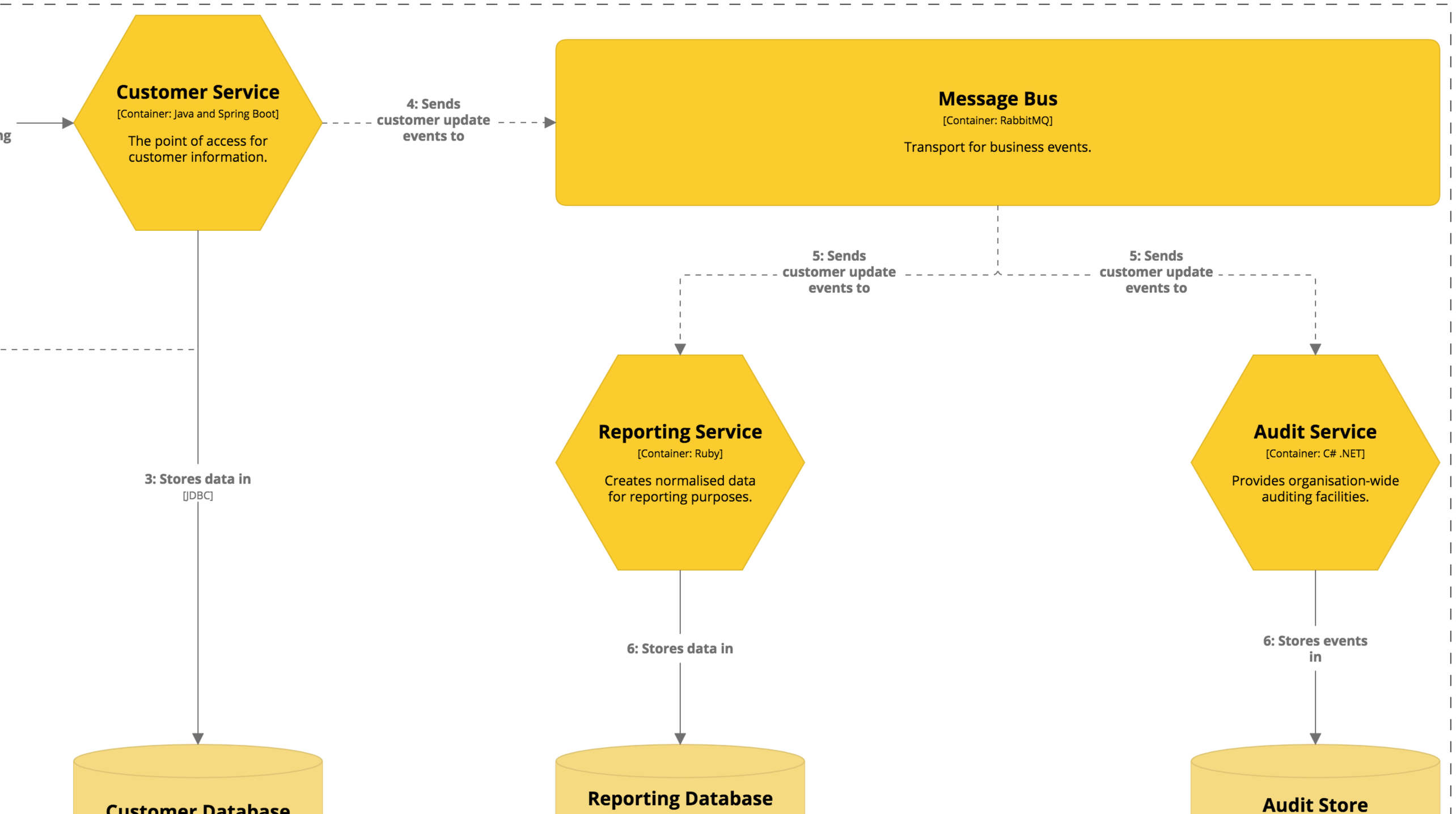

View list of vets
Clinic Employee
VetController
ClinicService
VetRepository
Relational Database
Requests list of vets from /vets
Calls findVets
Calls findAll
select * from vets
Clinic Employee
VetController
ClinicService
VetRepository
Relational Database
www.websequencediagrams.com




## Dynamic diagram

Shows how the "view list of vets" feature works.

Last modified: Monday 26 June 2017 12:59 UTC | Version: 95de1d9f8bf63560915331664b27a4a75ce1f1f6


Customer Service
[Container: Java and Spring Boot]
The point of access for customer information.
4: Sends customer update events to
Message Bus
[Container: RabbitMQ]
Transport for business events.
5: Sends customer update events to
5: Sends customer update events to
Reporting Service
[Container: Ruby]
Creates normalised data for reporting purposes.
Audit Service
[Container: C# .NET]
Provides organisation-wide auditing facilities.
3: Stores data in
[JDBC]
6: Stores data in
6: Stores events in
Reporting Database
Audit Store

Use dynamic diagrams to describe **patterns** or **complex interactions**

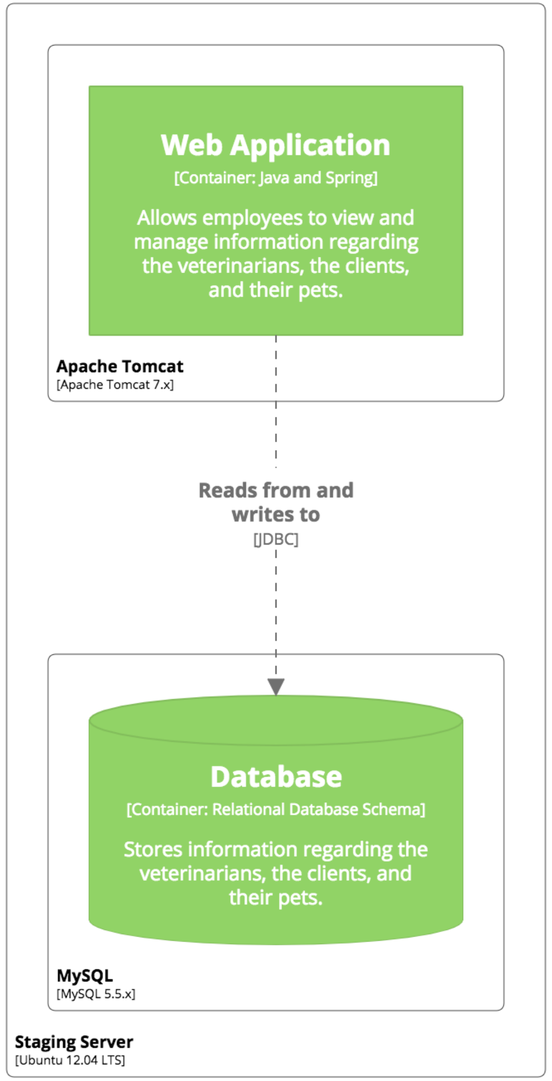
# Deployment diagrams

**Deployment** is about the mapping of containers to infrastructure

# Deployment Node

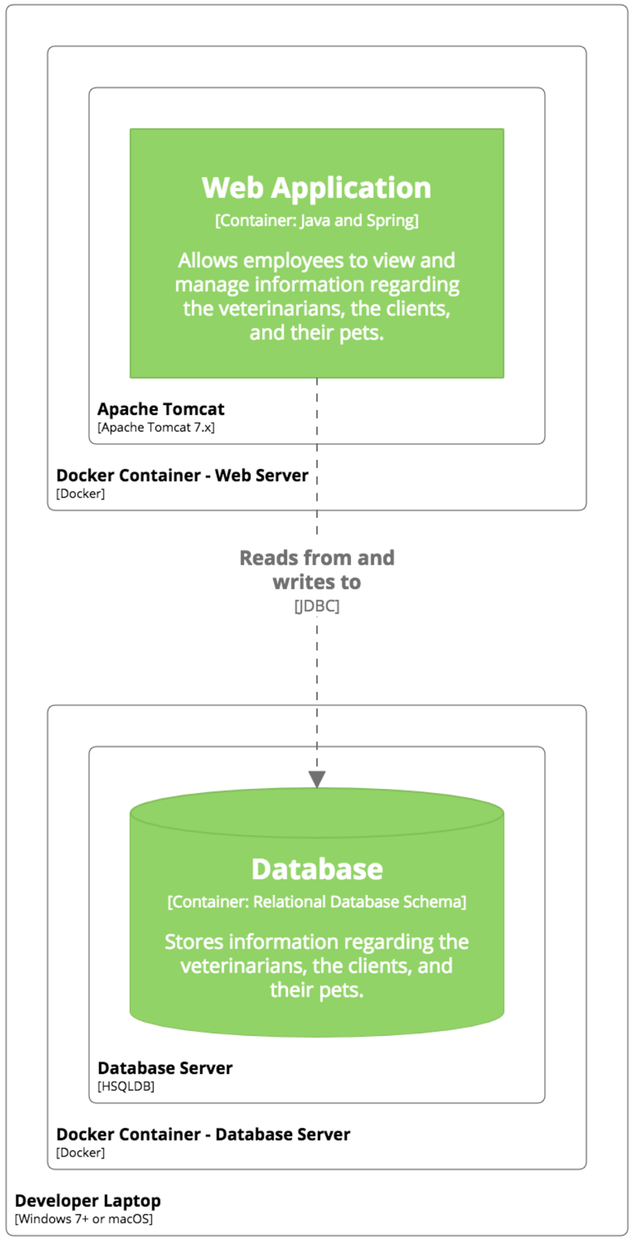
Physical infrastructure (a physical server or device), virtualised infrastructure (IaaS, PaaS, a virtual machine), containerised infrastructure (a Docker container), database server, Java EE web/application server, Microsoft IIS, etc

A deployment node can contain other **deployment nodes** or **container instances**



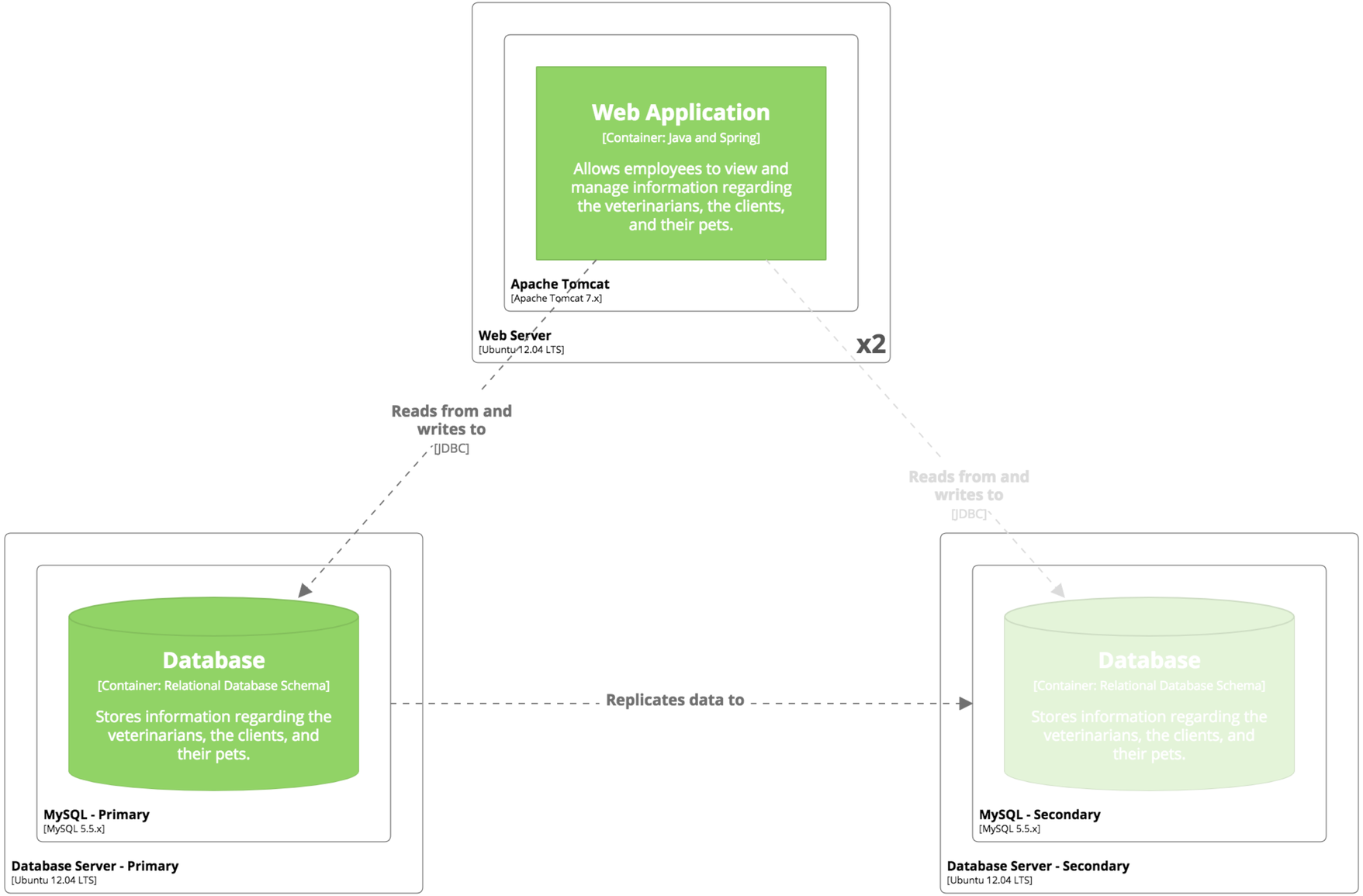

## Deployment diagram for Spring PetClinic

An example staging deployment scenario for the Spring PetClinic software system.
Last modified: Monday 26 June 2017 12:59 UTC | Version: 95de1d9f8bf63560915331664b27a4a75ce1f1f6




## Deployment diagram for Spring PetClinic

An example development deployment scenario for the Spring PetClinic software system.
Last modified: Monday 26 June 2017 12:59 UTC | Version: 95de1d9f8bf63560915331664b27a4a75ce1f1f6




## Deployment diagram for Spring PetClinic

An example live deployment scenario for the Spring PetClinic software system.
Last modified: Monday 26 June 2017 12:59 UTC | Version: 95de1d9f8bf63560915331664b27a4a75ce1f1f6

# Frequently asked questions

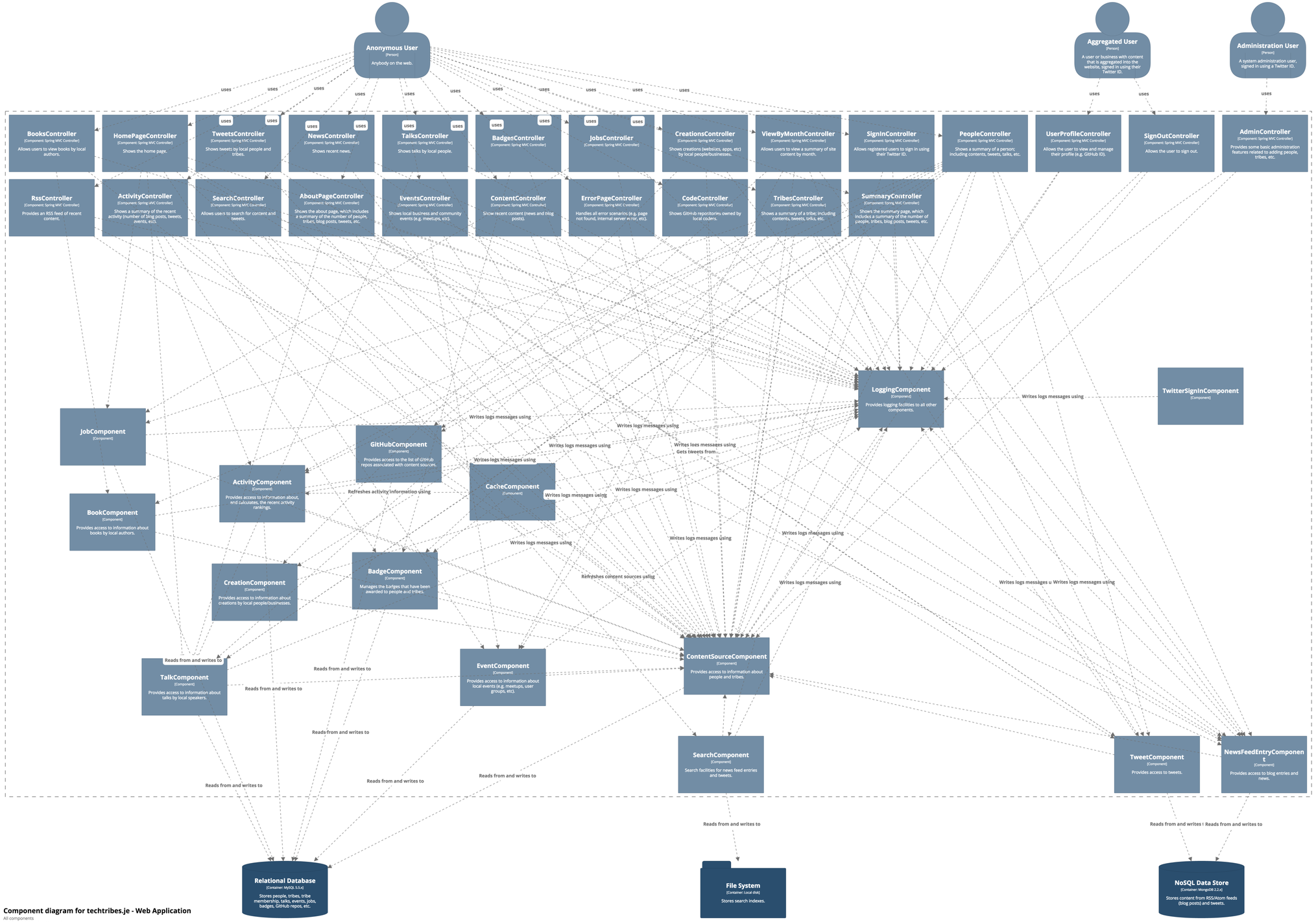
What's the inspiration behind the C4 model?

# Why "container"?

# Can we change the terminology?

How do you model microservices and serverless?

How do you diagram large and complex software systems?



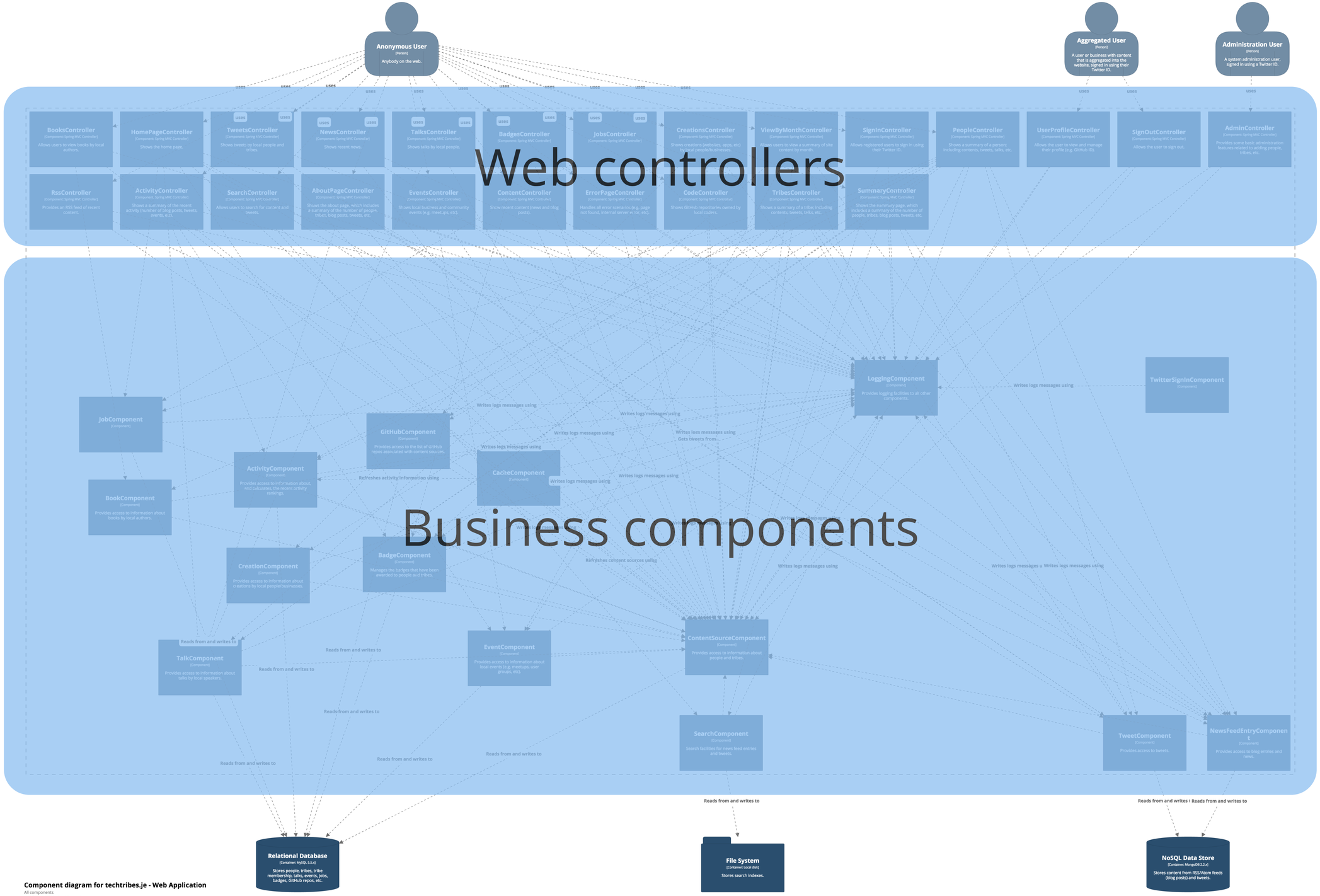

**Component diagram for techtribes.je - Web Application**

All components

Sunday 24 July 2016 04:59 UTC

# Option 1: Introduce additional abstractions

(nested components, sub-components, layers, etc)



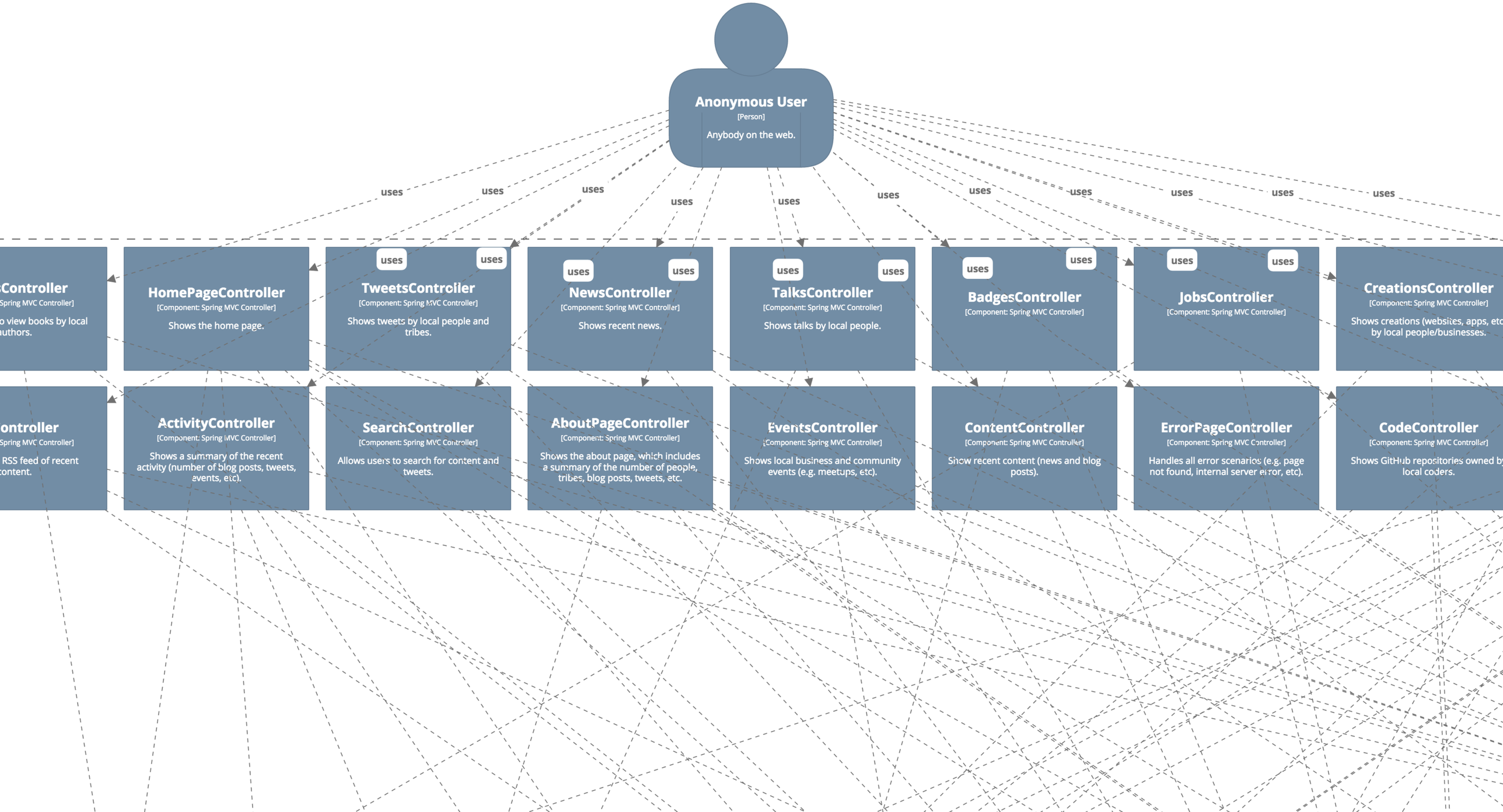

**Component diagram for techtribes.je - Web Application**

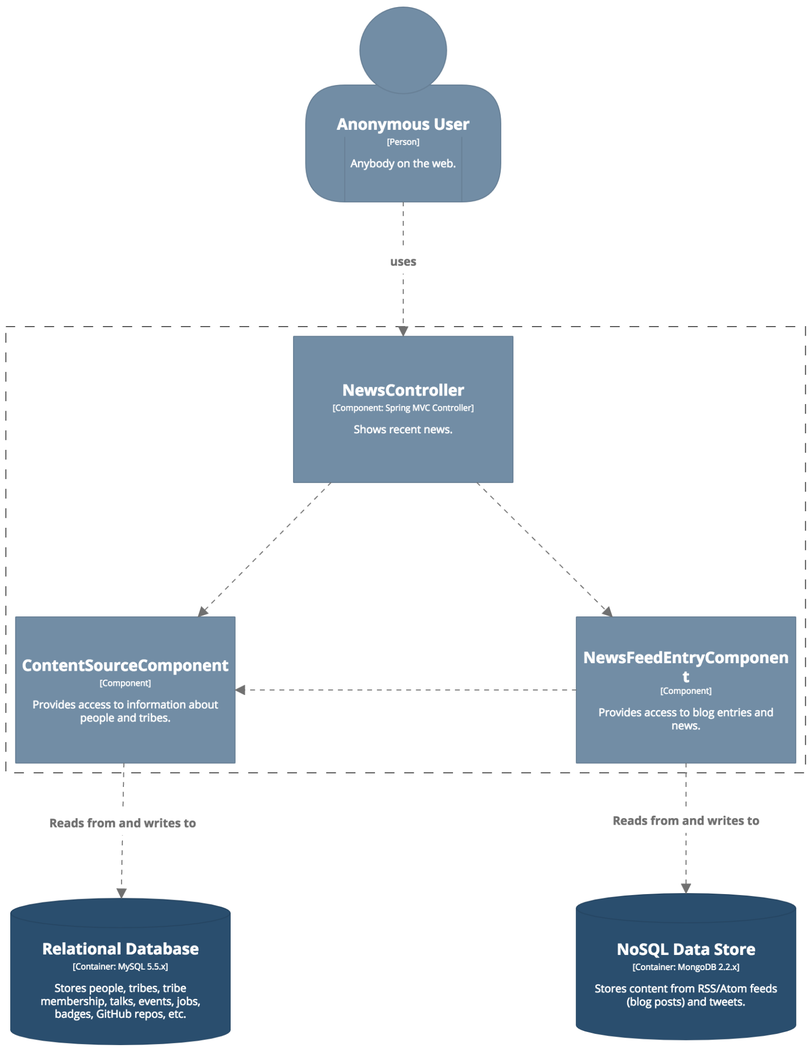
All components

Sunday 24 July 2016 04:59 UTC

# Option 2: Partition the diagrams

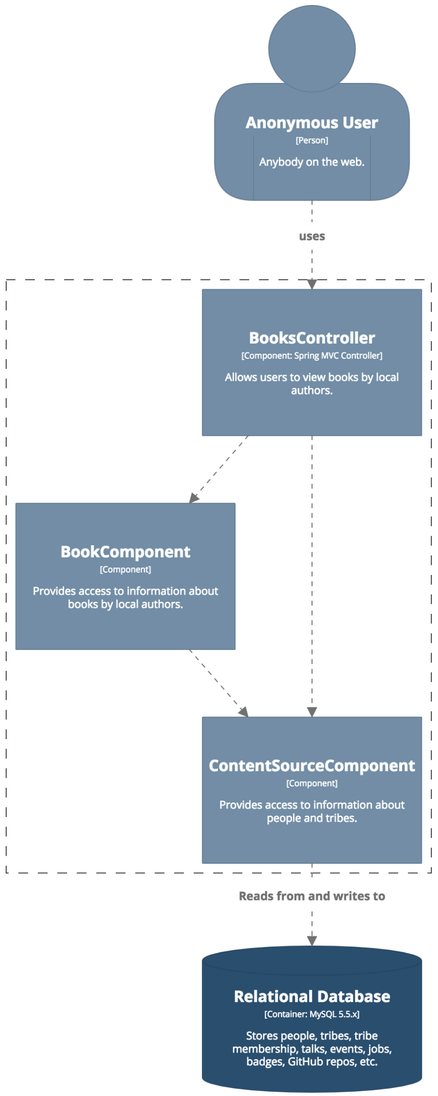
(one diagram per business concept, feature set, bounded context, aggregate root, vertical slice, etc)


Anonymous User
[Person]
Anybody on the web.
uses
Controller
Spring MVC Controller]
o view books by local authors.
HomePageController
[Component: Spring MVC Controller]
Shows the home page.
TweetsController
[Component: Spring MVC Controller]
Shows tweets by local people and tribes.
NewsController
[Component: Spring MVC Controller]
Shows recent news.
TalksController
[Component: Spring MVC Controller]
Shows talks by local people.
BadgesController
[Component: Spring MVC Controller]
JobsController
[Component: Spring MVC Controller]
CreationsController
[Component: Spring MVC Controller]
Shows creations (websites, apps, etc by local people/businesses.
ontroller
Spring MVC Controller]
RSS feed of recent content.
ActivityController
[Component: Spring MVC Controller]
Shows a summary of the recent activity (number of blog posts, tweets, events, etc).
SearchController
[Component: Spring MVC Controller]
Allows users to search for content and tweets.
AboutPageController
[Component: Spring MVC Controller]
Shows the about page, which includes a summary of the number of people, tribes, blog posts, tweets, etc.
EventsController
[Component: Spring MVC Controller]
Shows local business and community events (e.g. meetups, etc).
ContentController
[Component: Spring MVC Controller]
Show recent content (news and blog posts).
ErrorPageController
[Component: Spring MVC Controller]
Handles all error scenarios (e.g. page not found, internal server error, etc).
CodeController
[Component: Spring MVC Controller]
Shows GitHub repositories owned by local coders.



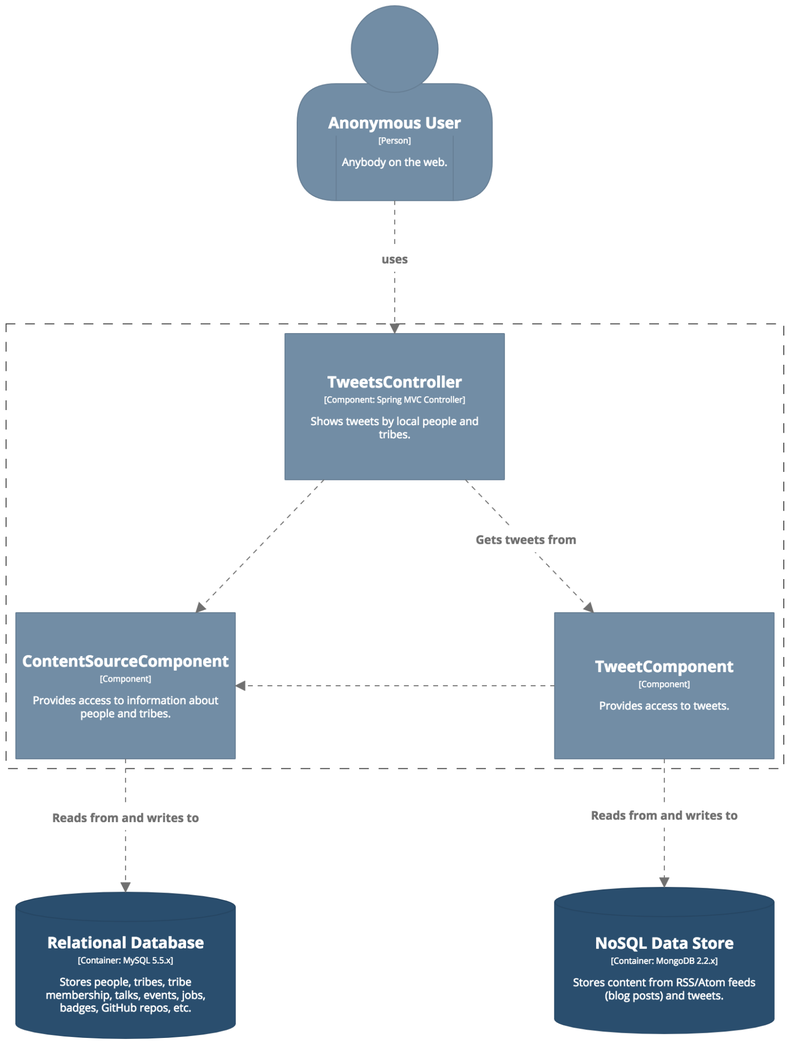

**Component diagram for techtribes.je - Web Application**

This diagram shows a slice through the web application, starting at NewsController.
Sunday 24 July 2016 04:59 UTC




**Component diagram for techtribes.je - Web Application**

This diagram shows a slice through the web application, starting at BooksController.
Sunday 24 July 2016 04:59 UTC




**Component diagram for techtribes.je - Web Application**

This diagram shows a slice through the web application, starting at TweetsController.
Sunday 24 July 2016 04:59 UTC

Will the diagrams become outdated quickly?

Why doesn't the C4 model cover business processes, workflows, state machines, domain models, data models, etc?

# The C4 model vs UML, ArchiMate and SysML?

# Can we combine C4 and arc42?

Does the C4 model imply a design process or team structure?

# Draw a System Context diagram to describe your software system

(people and software systems)

**30 minutes**

# Draw a Container diagram to describe your software system

(deployable/runnable units within the software system in scope)

**30 minutes**

# Draw System Context and Container diagrams to describe your software system

**60 minutes**

# Tooling

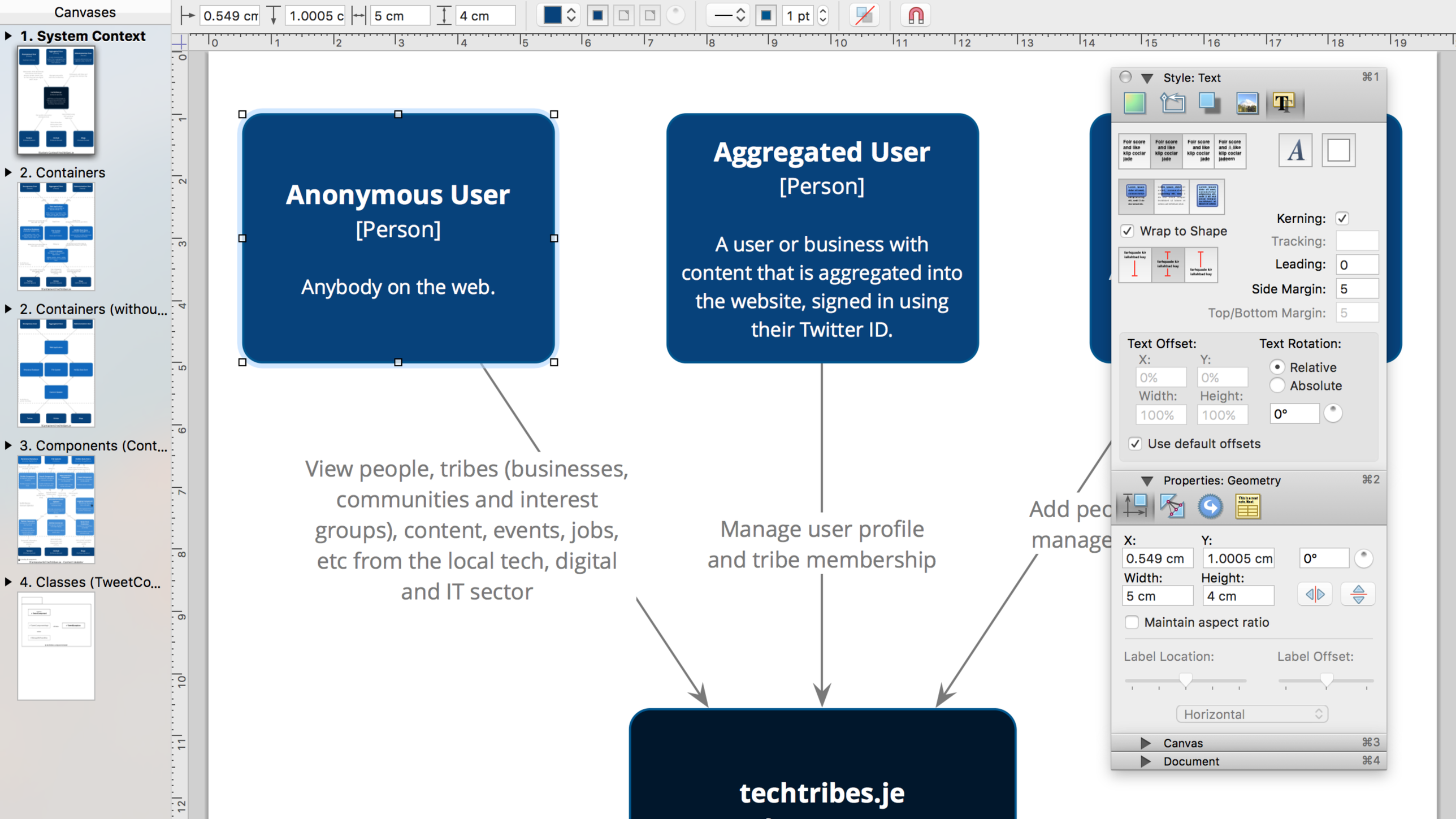
What **tools** do you recommend?

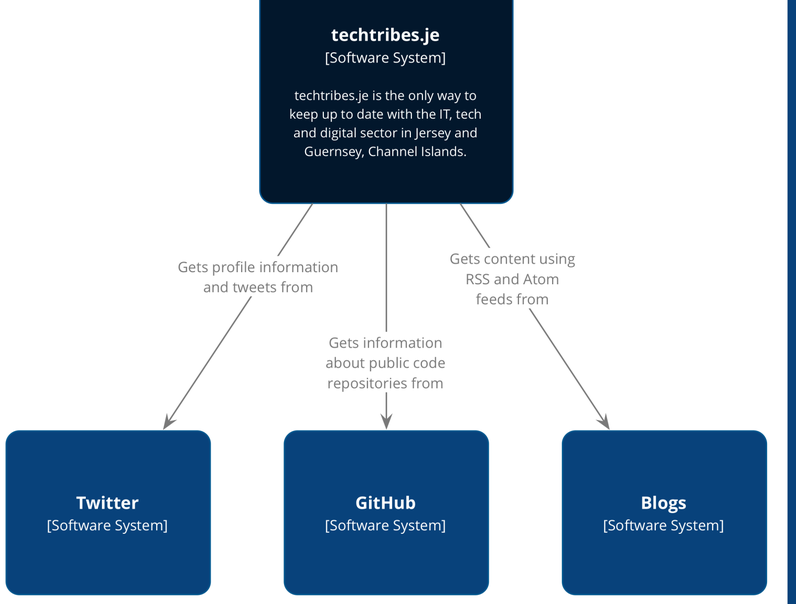
# Whiteboards and paper for up front design

General purpose diagramming tools allow you to draw **anything**

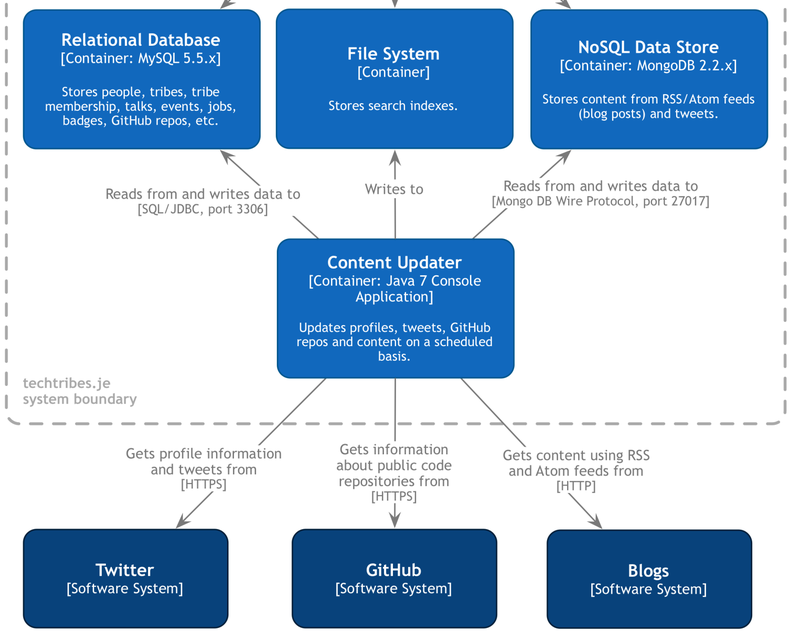
Microsoft Visio, OmniGraffle, Gliffy,

Lucidchart, draw.io, Creately, ...

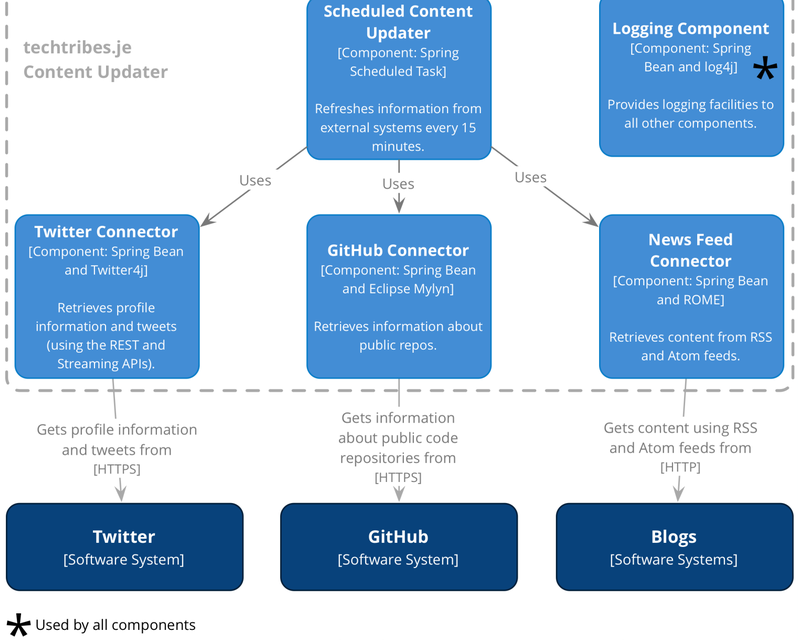

Canvases
1. System Context
2. Containers
2. Containers (withou...
3. Components (Cont...
4. Classes (TweetCo...
0.549 cm
1.0005 c
5 cm
4 cm
1 pt
Anonymous User
[Person]
Anybody on the web.
Aggregated User
[Person]
A user or business with content that is aggregated into the website, signed in using their Twitter ID.
View people, tribes (businesses, communities and interest groups), content, events, jobs, etc from the local tech, digital and IT sector
Manage user profile and tribe membership
Add pec
manage
techtribes.je
Style: Text
Wrap to Shape
Kerning:
Tracking:
Leading: 0
Side Margin: 5
Top/Bottom Margin: 5
Text Offset:
X: 0%
Y: 0%
Width: 100%
Height: 100%
Text Rotation:
Relative
Absolute
0°
Use default offsets
Properties: Geometry
X: 0.549 cm
Y: 1.0005 cm
0°
Width: 5 cm
Height: 4 cm
Maintain aspect ratio
Label Location:
Label Offset:
Horizontal
Canvas
Document




[System Context] techtribes.je




[Containers] techtribes.je




[Components] techtribes.je - Content Updater

# Static diagrams need to be kept up to date **individually**

# Diagrams vs models

Text-based tools allow you to

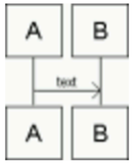
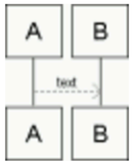
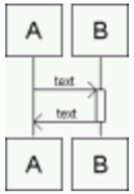
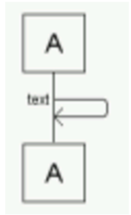
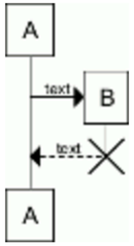
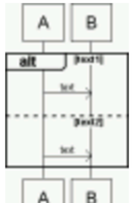
**create diagrams using text**

PlantUML, WebSequenceDiagrams, yUML, nomnoml, ...

# WebSequenceDiagrams

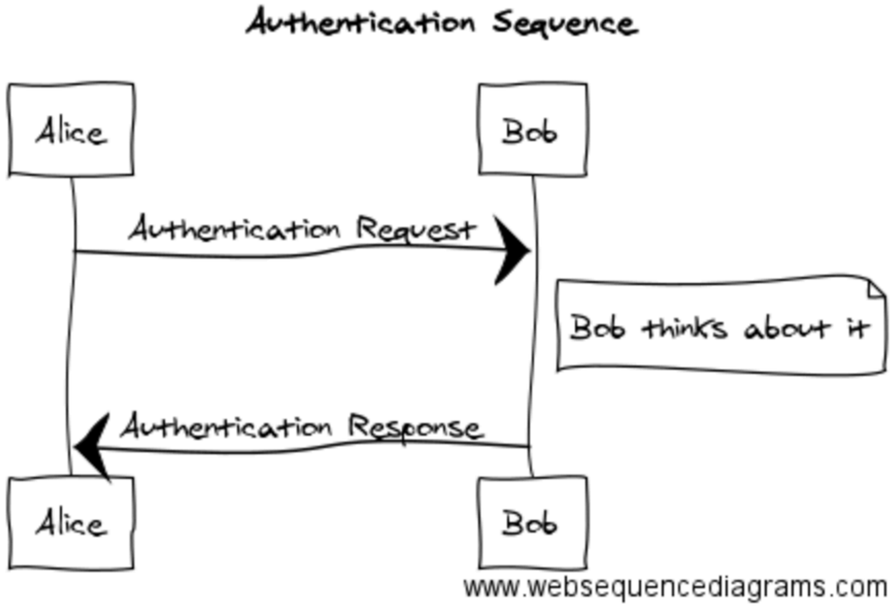
Sign in with Google

Account Styles Share

Not saved

```
title Authentication Sequence

Alice->Bob: Authentication Request
note right of Bob: Bob thinks about it
Bob->Alice: Authentication Response
```



# Structurizr Express

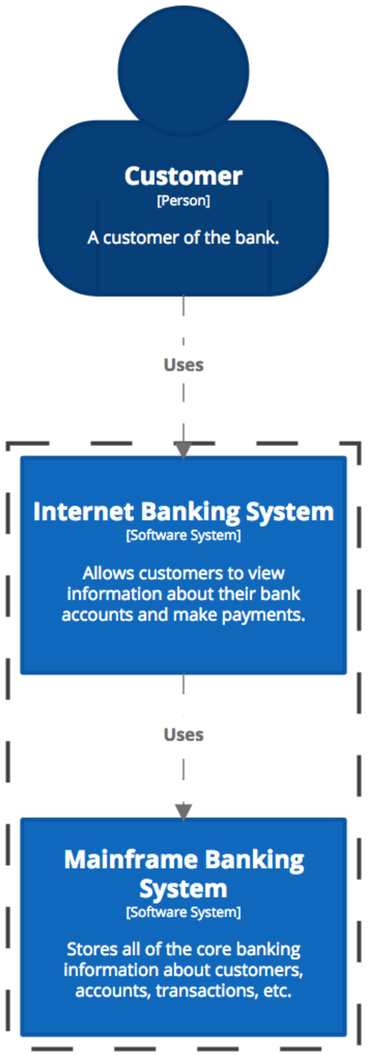
Welcome to Structurizr Express, a tool to help you quickly create a single System Landscape, System Context, Container or Component diagram from the C4 model by defining it as text. See the help page for more information.

If you would like to create a full software architecture model with multiple diagrams, take a look at the Structurizr client libraries for Java and .NET instead.

JSON | YAML | Diagram | Elements | Relationships | Styles


```
{
  "type": "System Context",
  "scope": "Internet Banking System",
  "description": "The system context diagram for the
Internet Banking System.",
  "size": "A5_Landscape",


  "elements": [
    {
      "type": "Person",
      "name": "Customer",
      "description": "A customer of the bank.",
      "tags": "",
      "position": "1040,97"
    },
    {
      "type": "Software System",
      "name": "Internet Banking System",
      "description": "Allows customers to view information
about their bank accounts and make payments.",
      "tags": "Internal",
      "position": "1015,725",
      "containers": []
    },
    {
      "type": "Software System",
      "name": "Mainframe Banking System",
      "description": "Stores all of the core banking
```


A5 - Landscape




**System Context diagram for Internet Banking System**

The system context diagram for the Internet Banking System.

Last modified: Thursday 18 January 2018 11:20 UTC

Text-based diagrams
can be **version controlled**,
generated from build scripts, etc

Text-based diagrams can’t be **queried**

**Architecture description languages**
provide a way to describe
architecture as text

Darwin, ACME, Koala, Wright, ...

# Modelling tools allow you to create a **model** of your software system

Sparx Enterprise Architect, IBM Rational Software Architect, StarUML, GenMyModel, Visual Paradigm, ...

Working Diagrams

1. Spring PetClinic - System C

3. Spring PetClinic - Web App

Toolbox

Component

Component

Artifact

Interface

Frame

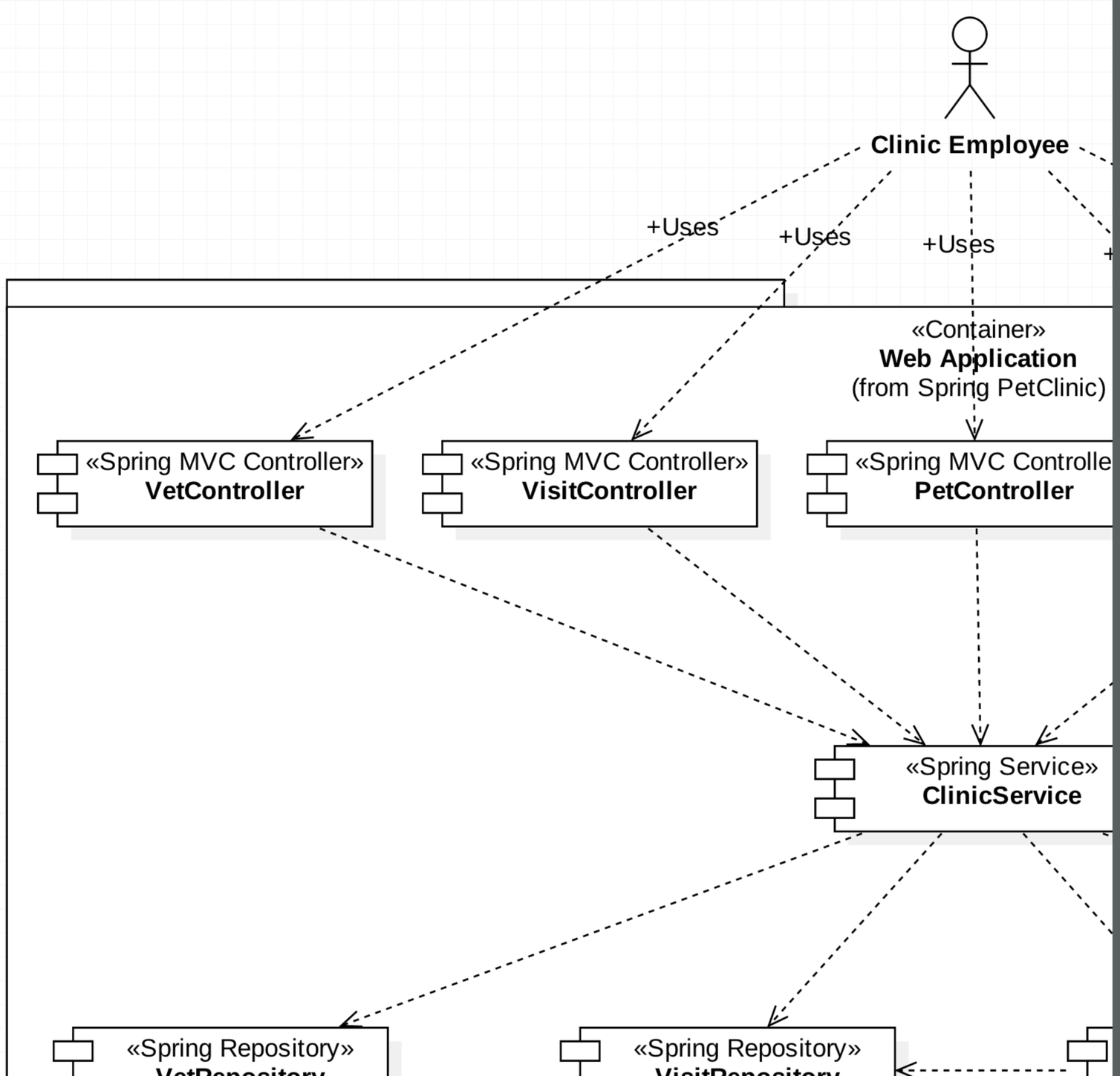
Dependency

Interface Realization

Component Realization

Deployment

Node




Model Explorer

- Spring PetClinic
  - Use Case Model
  - Analysis Model
    - Analysis
  - Design Model
    - Classes
  - Implementation Model
    - 1. Spring PetClinic - System Context
    - 2. Spring PetClinic - Containers
    - 3. Spring PetClinic - Web Applicaation - C
    - «Software System»Spring PetClinic
      - «Container»Web Application
        - «Spring MVC Controller»VetContro
        - «Spring MVC Controller»VisitContr
        - «Spring MVC Controller»PetContro
        - «Spring MVC Controller»OwnerCo
        - «Spring Service»ClinicService
        - «Spring MVC Controller»CrashCon
        - «Spring Repository»OwnerReposit

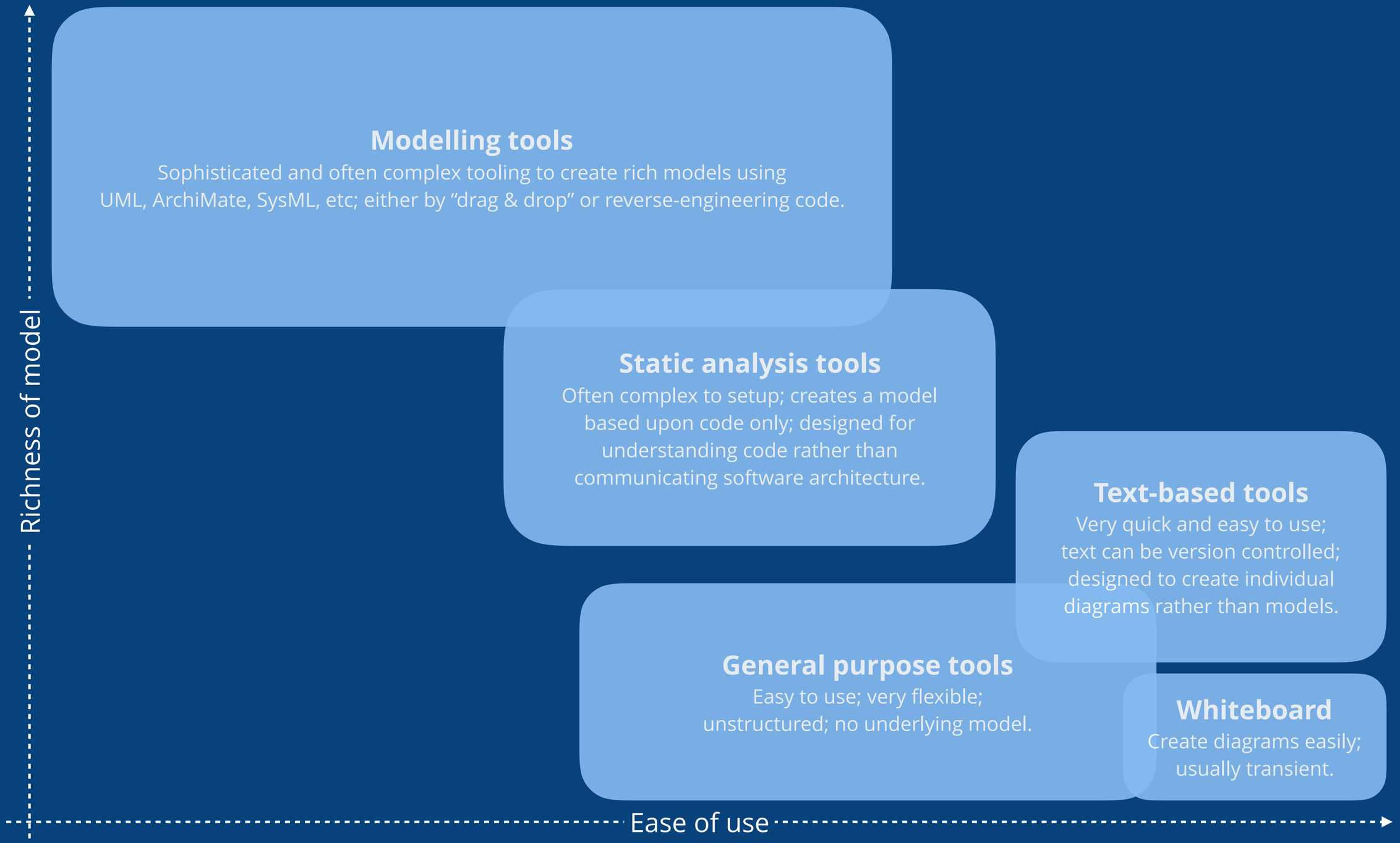
Editors

Properties

| | |
|---|---|
| name | VetController |
| stereotype | Spring MVC Controlle |
| visibility | public |
| isAbstract | ☐ |
| isFinalSpecialization | ☐ |
| isLeaf | ☐ |
| isIndirectlyInstantiated | ☑ |

Static analysis and dependency management tools can help **understand code structure**

Structure101, Lattix, JArchitect, NDepend, JDepend, jQAssistant, Degraph, ...


Richness of model
Modelling tools
Sophisticated and often complex tooling to create rich models using UML, ArchiMate, SysML, etc; either by “drag & drop” or reverse-engineering code.
Static analysis tools
Often complex to setup; creates a model based upon code only; designed for understanding code rather than communicating software architecture.
Text-based tools
Very quick and easy to use; text can be version controlled; designed to create individual diagrams rather than models.
General purpose tools
Easy to use; very flexible; unstructured; no underlying model.
Whiteboard
Create diagrams easily; usually transient.
Ease of use

## What do you see as the future of software architecture documentation?

**Eoin**: I hope that in the future we'll need very little software architecture documentation because we'll be able to see the architecture in the code and the running system! One of the reasons we need much of our architecture documentation today is because there's no way of representing architectural structures directly using the technologies we have at our disposal. I'd love to see our architectural constructs as first class implementation structures and then architecture documentation can evolve to capture decisions, rationale and analysis, rather than just capturing structures. On the way to this nirvana, I hope that work going on in the areas of DSLs and ADLs (architecture description languages) point the more immediate way forward, as we improve our description languages, on the way to working out how to embed the information right in the running system.

**Paulo**: The software architecture discipline is fairly new. There is a long path ahead until we get to a point where an architect creates architecture documentation that is readily understood by a developer who has never worked with that architect. The way to get there is to let new architects learn software architecture at school rather than try-and-error in the battlefield. This education includes proper ways to represent the software architecture for other people's consumption. Important initiatives in the direction of good software architecture education are: the work of Grady Booch on the handbook of software architecture and the publications and curriculum developed at the SEI.

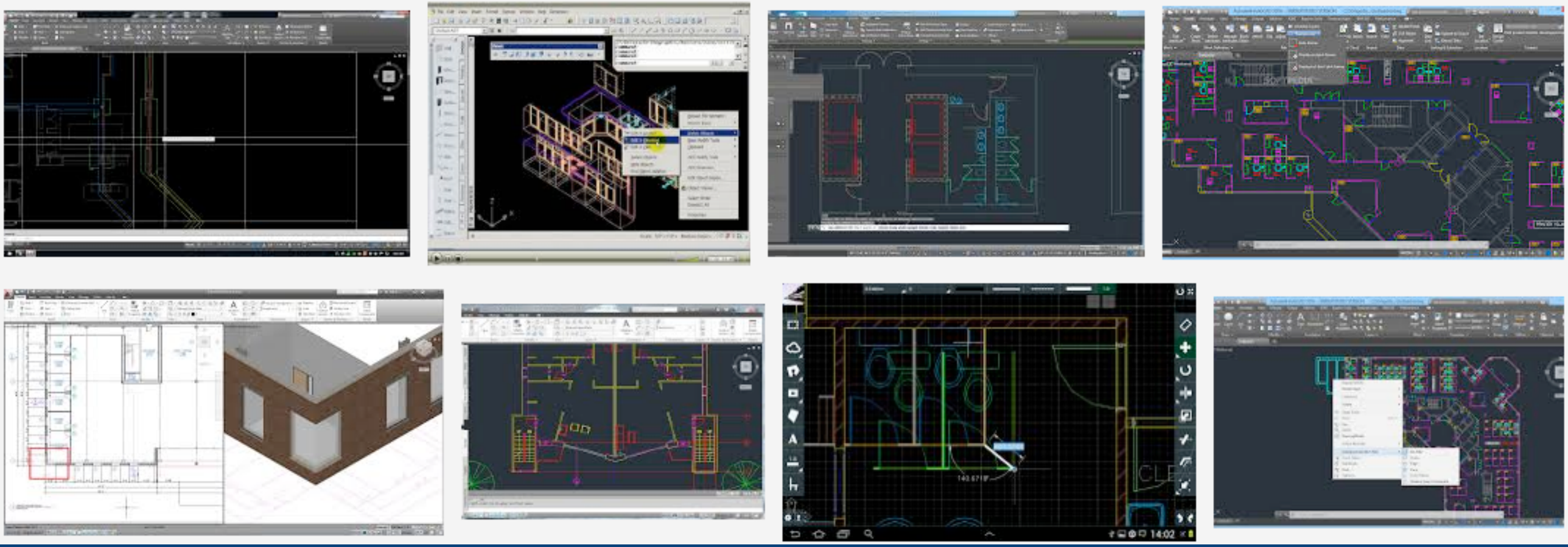
**Grady**: There is a lot of energy being applied today with regard to architectural frameworks and methods: TOGAF, NEA, DoDAF, MoDAF, FSAM, Zachman, and so on. The good news is that there is a vibrant dialog going on with regard to these frameworks and methods - but I expect there will be a shakeout/simplification over time.

**Len**: The ideal development environment is one for which the documentation is available for essentially free with the push of a button. This will require an integrated development, requirements management, and project management environment. Although this will be a long time coming, it provides a worthy goal to strive for.

# Virtual Panel on Software Architecture Documentation

(2009)

http://www.infoq.com/articles/virtual-panel-arch-documentation



# Do structural engineers and building architects use general purpose drawing tools?

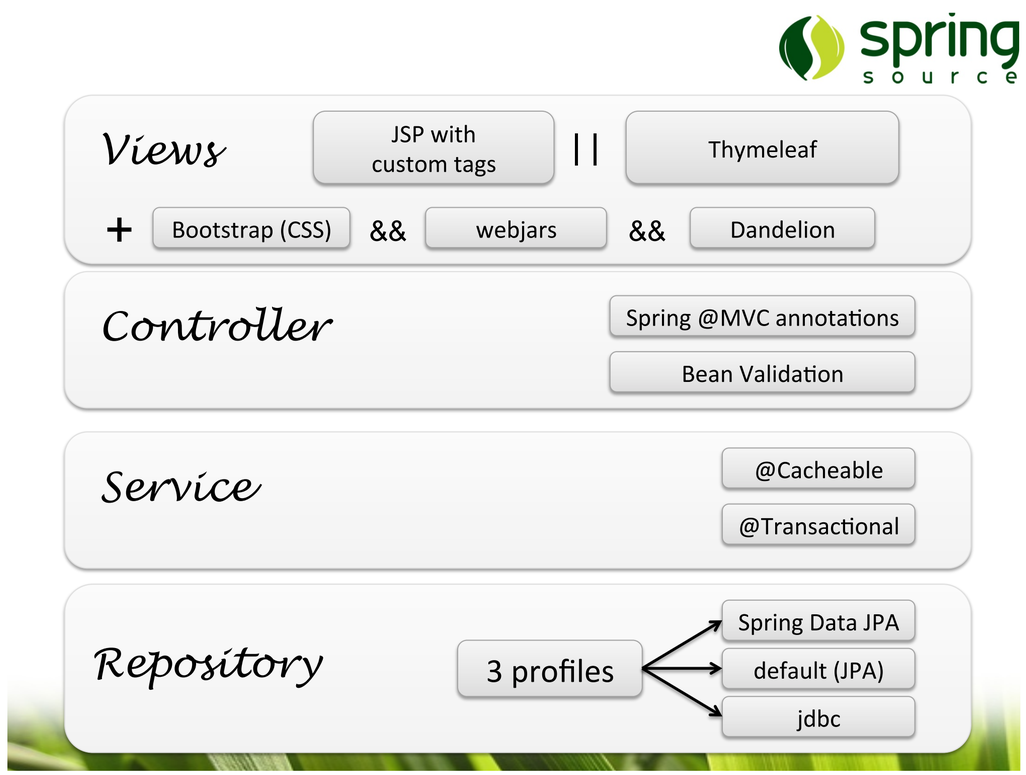
Diagrams are not useful for architectural improvement if they are not **connected to the code**

**Reverse-engineer**
code to diagrams?

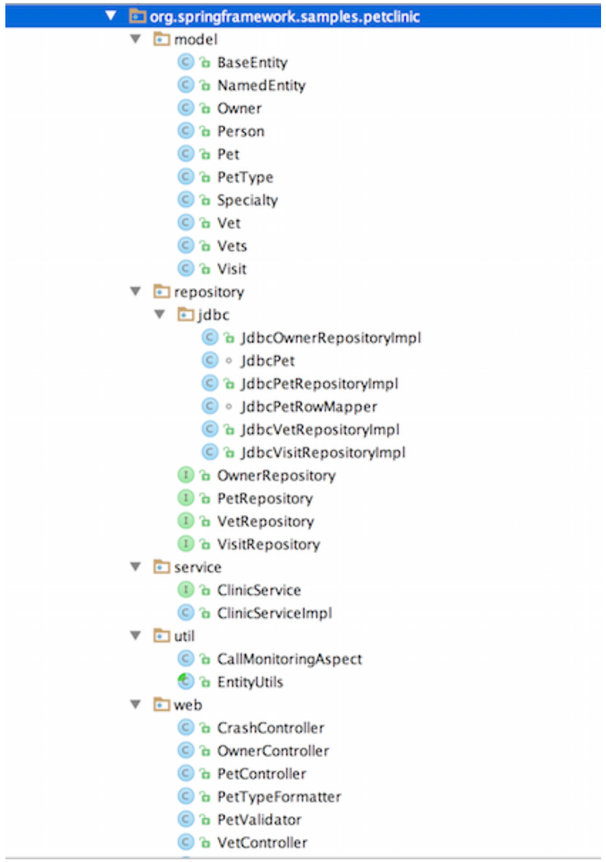
# Spring PetClinic

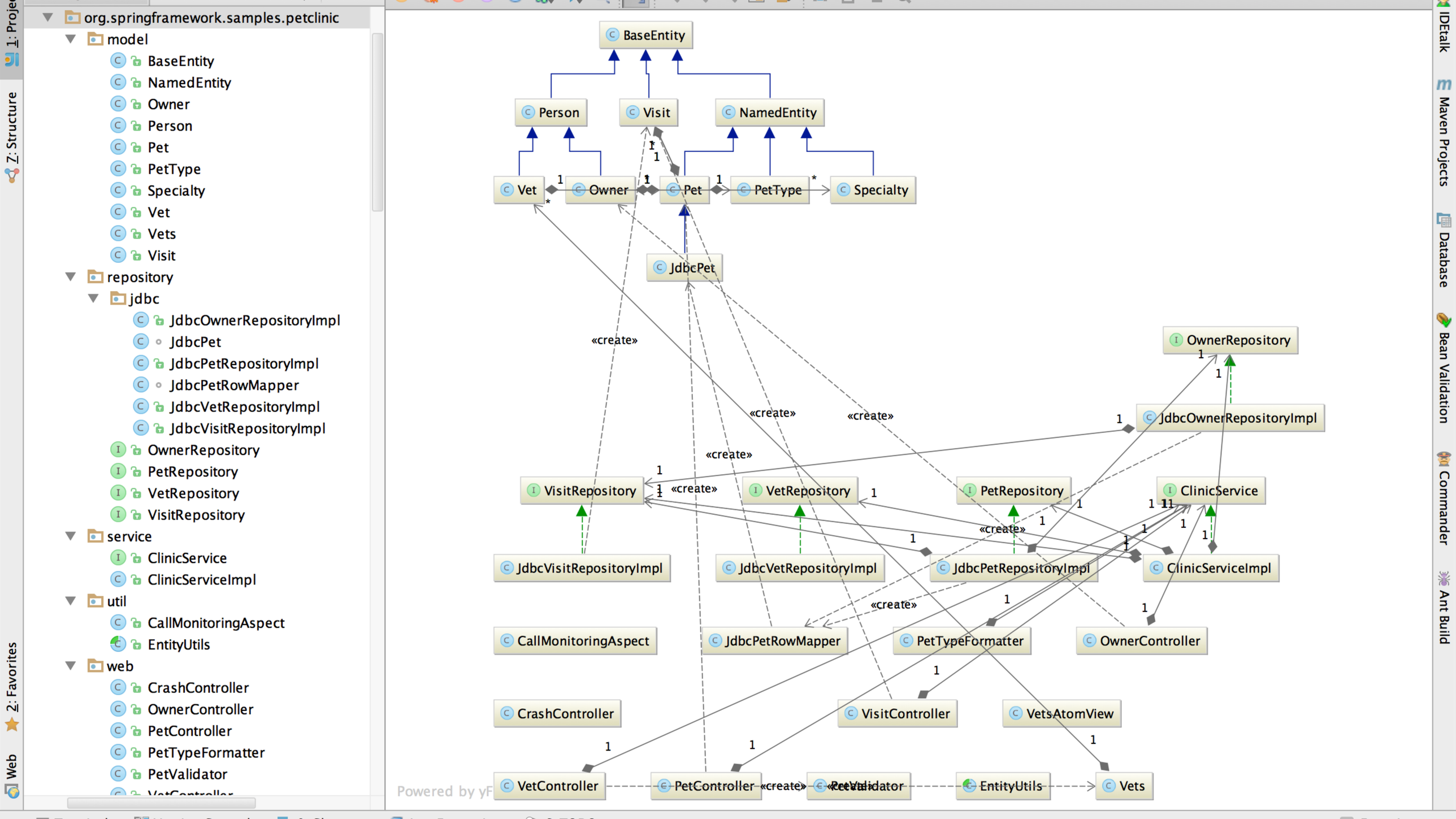
A sample application that illustrates how to build Java web applications using the Spring MVC framework

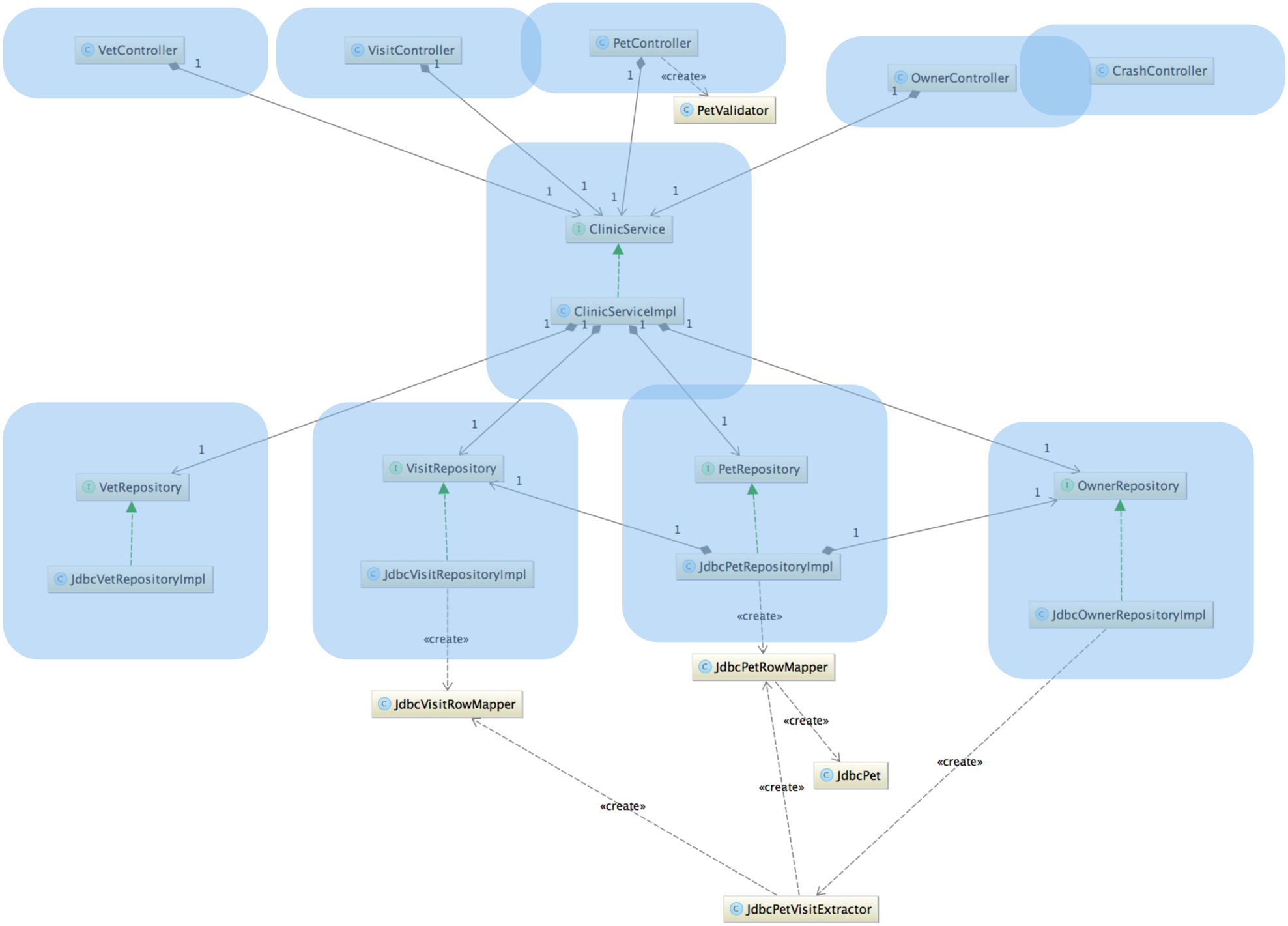
https://github.com/spring-projects/spring-petclinic/

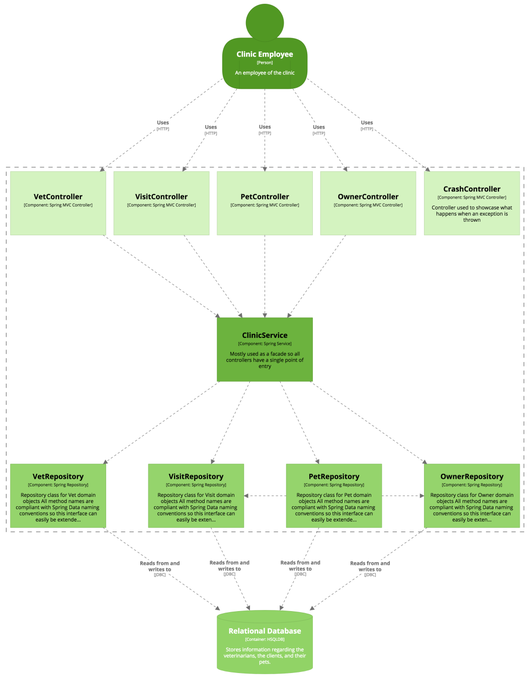



https://speakerdeck.com/michaelisvy/spring-petclinic-sample-application



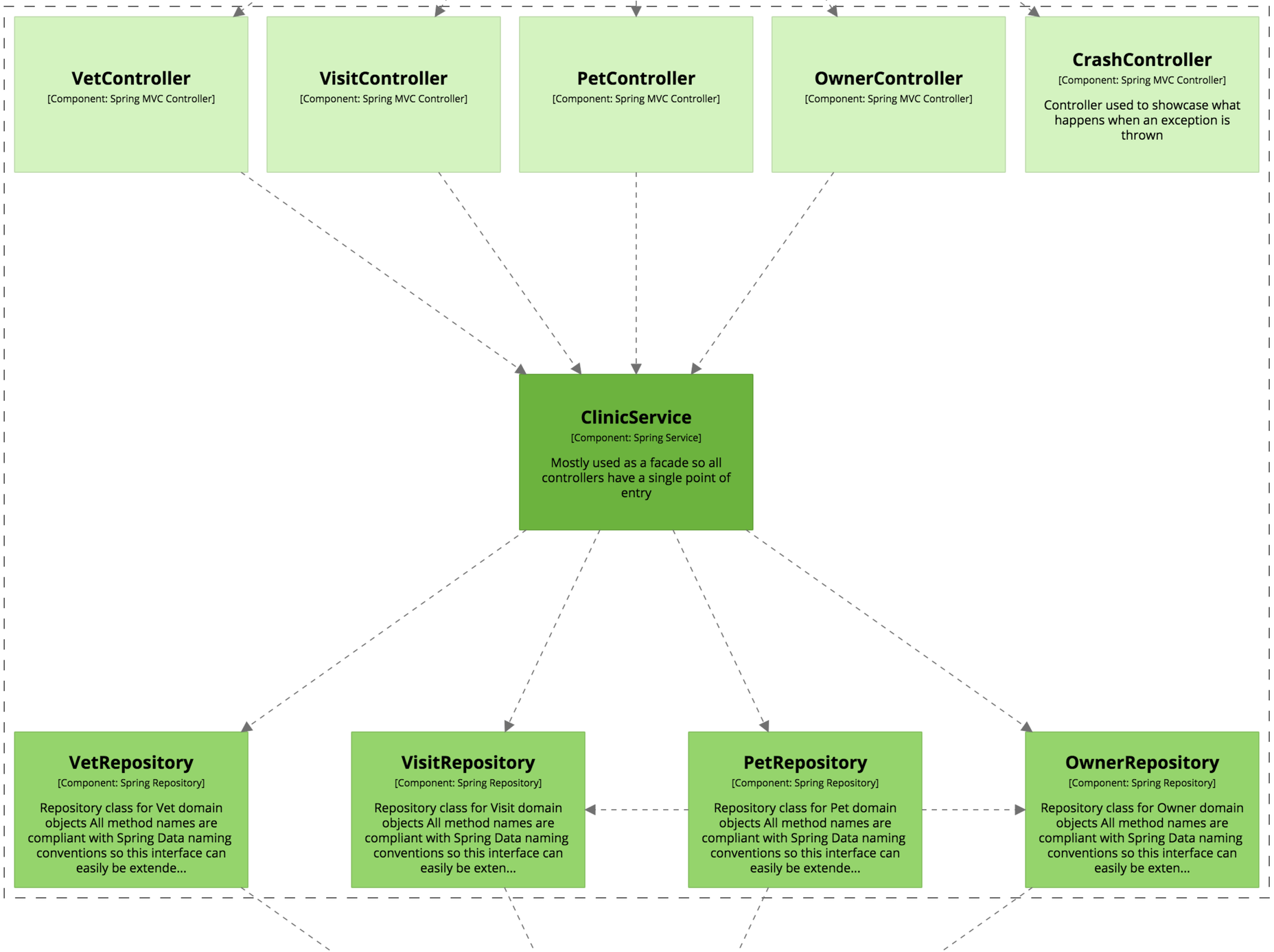

org.springframework.samples.petclinic
model
BaseEntity
NamedEntity
Owner
Person
Pet
PetType
Specialty
Vet
Vets
Visit
repository
jdbc
JdbcOwnerRepositoryImpl
JdbcPet
JdbcPetRepositoryImpl
JdbcPetRowMapper
JdbcVetRepositoryImpl
JdbcVisitRepositoryImpl
OwnerRepository
PetRepository
VetRepository
VisitRepository
service
ClinicService
ClinicServiceImpl
util
CallMonitoringAspect
EntityUtils
web
CrashController
OwnerController
PetController
PetTypeFormatter
PetValidator
BaseEntity
Person
Visit
NamedEntity
Vet
Owner
Pet
PetType
Specialty
JdbcPet
«create»
OwnerRepository
JdbcOwnerRepositoryImpl
VisitRepository
VetRepository
PetRepository
ClinicService
JdbcVisitRepositoryImpl
JdbcVetRepositoryImpl
JdbcPetRepositoryImpl
ClinicServiceImpl
CallMonitoringAspect
JdbcPetRowMapper
PetTypeFormatter
OwnerController
CrashController
VisitController
VetsAtomView
VetController
PetController
EntityUtils
Vets
Powered by yF


VetController
VisitController
PetController
«create»
PetValidator
OwnerController
CrashController
ClinicService
ClinicServiceImpl
VetRepository
JdbcVetRepositoryImpl
VisitRepository
JdbcVisitRepositoryImpl
PetRepository
JdbcPetRepositoryImpl
OwnerRepository
JdbcOwnerRepositoryImpl
JdbcPetRowMapper
JdbcVisitRowMapper
JdbcPet
JdbcPetVisitExtractor



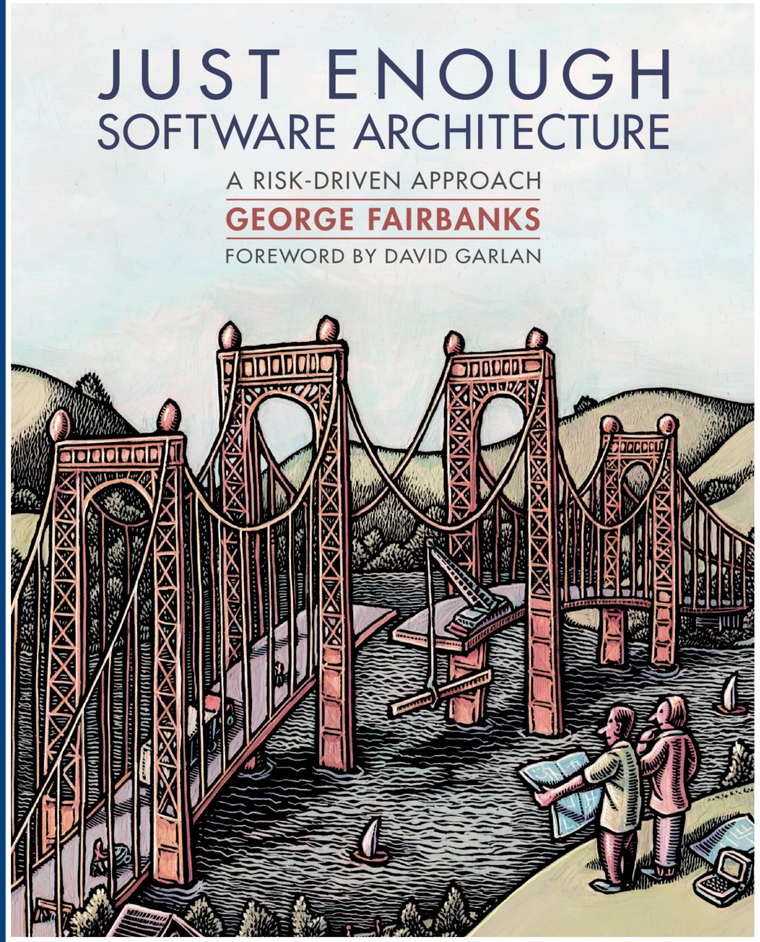

**Component diagram for Spring PetClinic - Web Application**

The Components diagram for the Spring PetClinic web application.
Friday 18 November 2016 22:21 UTC



Most tools see code,
not **components**

Information about software architecture doesn’t exist in the code




**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.

# “architecturally-evident coding style”

# Examples of architecturally-evident coding styles

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

Naming conventions (*Service)

Namespacing/packaging (com.mycompany.system.components.*)

Maven & Gradle modules, OSGi & Java 9 modules

JavaScript module patterns, ECMAScript 6 modules, microservices, etc

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

# **Executable** architecture description language

Structurizr for Java and .NET

Richness of model

## Modelling tools

Sophisticated and often complex tooling to create rich models using UML, ArchiMate, SysML, etc; either by “drag & drop” or reverse-engineering code.

## Static analysis tools

Often complex to setup; creates a model based upon code only; designed for understanding code rather than communicating software architecture.

Structurizr

Visualise, document and explore your software architecture

Creates a relatively rich yet simple model, in a developer-friendly way using Java/C# code, supplemented with Markdown/AsciiDoc documentation.

## Text-based tools

Very quick and easy to use; text can be version controlled; designed to create individual diagrams rather than models.

## General purpose tools

Easy to use; very flexible; unstructured; no underlying model.

## Whiteboard

Create diagrams easily; usually transient.

Ease of use

```
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-projects/spring-petclinic/)");

    Model model = workspace.getModel();





}
```

```
// software systems and people
SoftwareSystem springPetClinic = model.addSoftwareSystem(
    "Spring PetClinic",
    "Allows employees to view and manage information regarding the
     veterinarians, the clients, and their pets.");

Person clinicEmployee = model.addPerson(
    "Clinic Employee", "An employee of the clinic");

clinicEmployee.uses(springPetClinic, "Uses");
```

```
// containers
Container webApplication = springPetClinic.addContainer(
    "Web Application",
    "Allows employees to view and manage information regarding the
     veterinarians, the clients, and their pets.",
    "Apache Tomcat 7.x");

Container relationalDatabase = springPetClinic.addContainer(
    "Relational Database",
    "Stores information regarding the veterinarians, the clients,
     and their pets.", "HSQLDB");

clinicEmployee.uses(webApplication,
    "Uses", “HTTP");

webApplication.uses(relationalDatabase,
    "Reads from and writes to", "JDBC, port 9001");
```

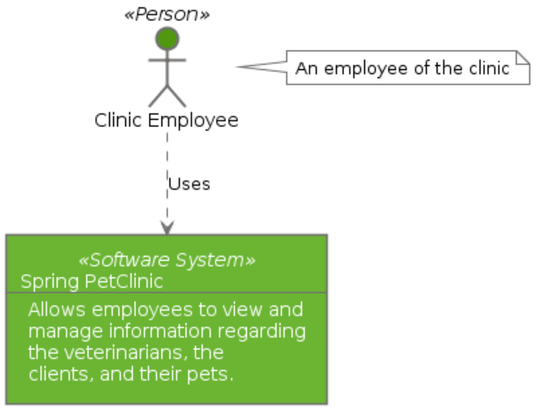
```java
// components
ComponentFinder componentFinder = new ComponentFinder(
    webApplication,
    "org.springframework.samples.petclinic",
    new SpringComponentFinderStrategy(
        new ReferencedTypesSupportingTypesStrategy()
    ),
    new SourceCodeComponentFinderStrategy(
        new File(sourceRoot, "/src/main/java/"), 150));

componentFinder.findComponents();
```

```
// connect components with other model elements
webApplication.getComponents().stream()
    .filter(c -> c.getTechnology().equals(SpringComponentFinderStrategy.SPRING_MVC_CONTROLLER))
    .forEach(c -> clinicEmployee.uses(c, "Uses", "HTTP"));

webApplication.getComponents().stream()
    .filter(c -> c.getTechnology().equals(SpringComponentFinderStrategy.SPRING_REPOSITORY))
    .forEach(c -> c.uses(relationalDatabase, "Reads from and writes to", "JDBC"));
```

```
// system context, container and component views
ViewSet viewSet = workspace.getViews();

SystemContextView contextView = viewSet.createContextView(
    springPetClinic, "context", "Context view for Spring PetClinic");
contextView.addAllSoftwareSystems();
contextView.addAllPeople();

ContainerView containerView = viewSet.createContainerView(
    springPetClinic, "containers", "Container view for Spring PetClinic");
containerView.addAllPeople();
containerView.addAllSoftwareSystems();
containerView.addAllContainers();

ComponentView componentView = viewSet.createComponentView(
    webApplication, "components", "Component view for the Spring PetClinic webapp.");
componentView.addAllComponents();
componentView.addAllPeople();
componentView.add(relationalDatabase);
```

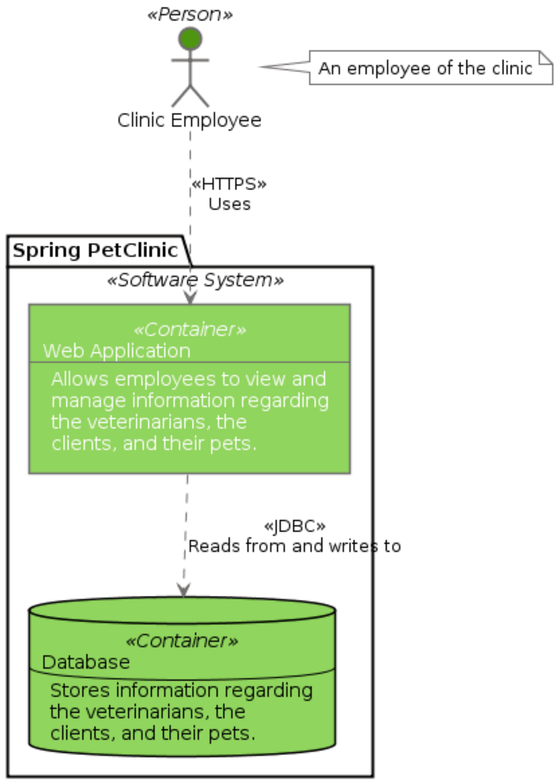
You can create
many **visualisations**
from a single model

## Spring PetClinic - System Context



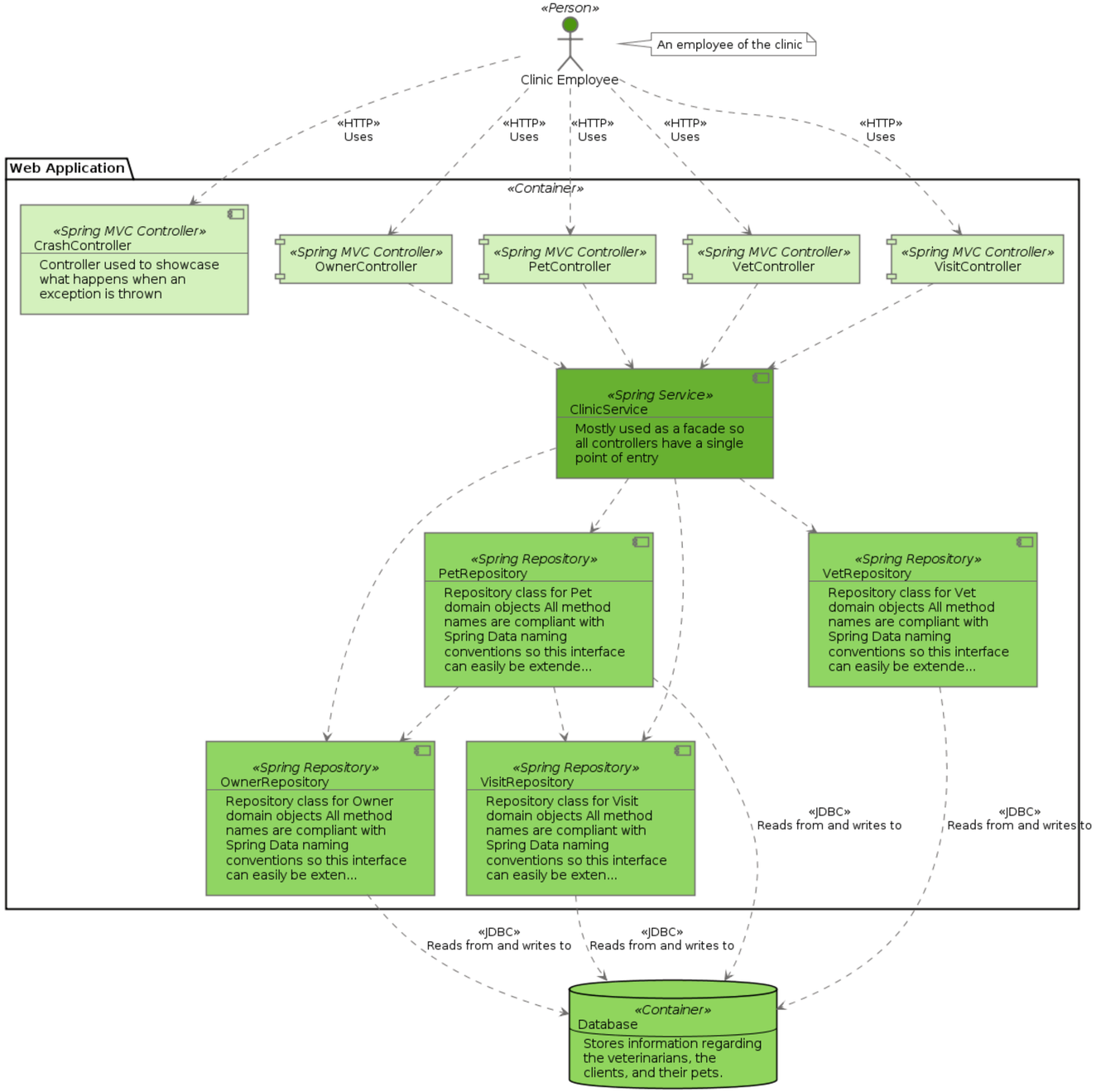

The System Context diagram for the Spring PetClinic system.

## Spring PetClinic - Containers

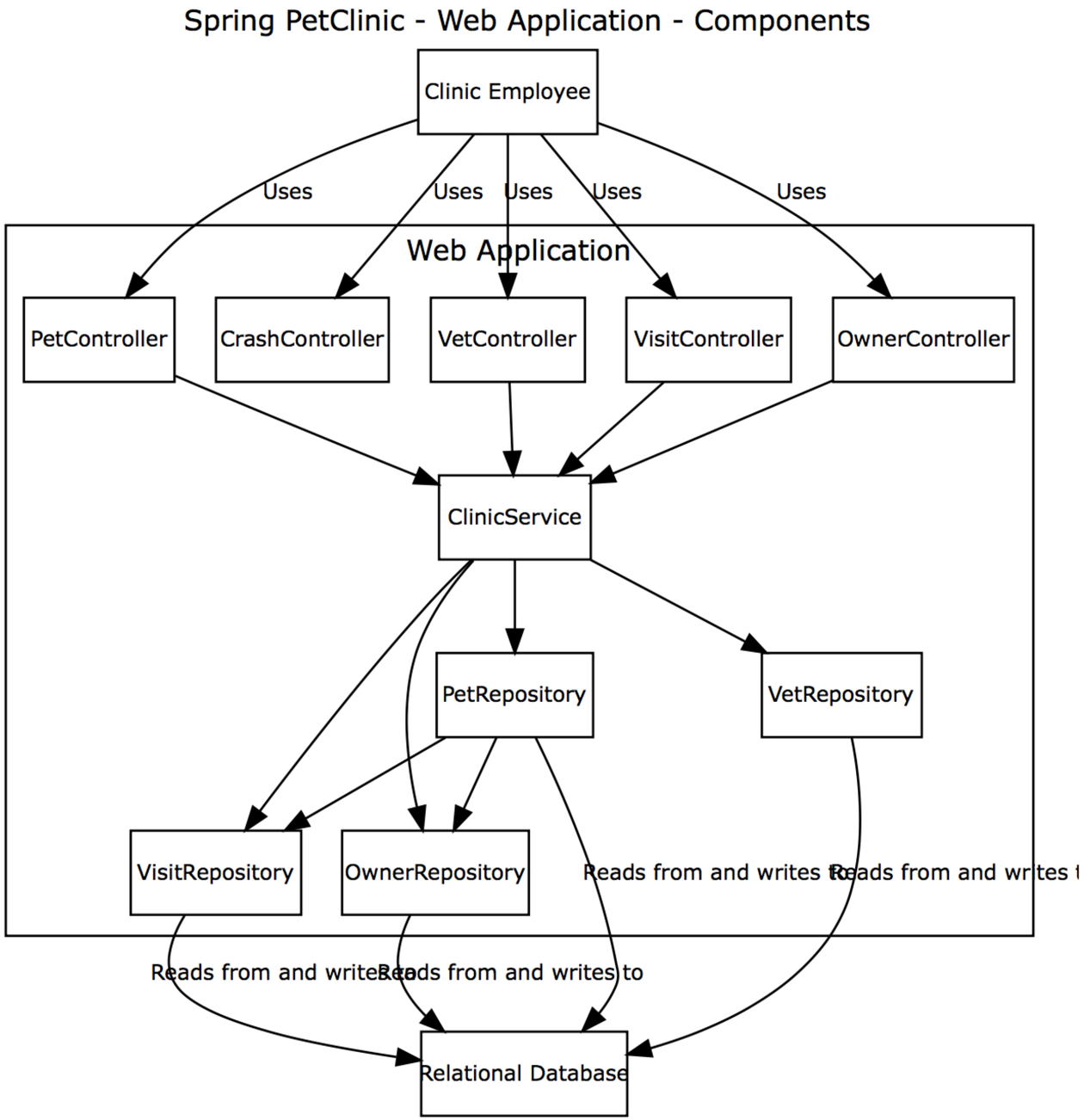



The Containers diagram for the Spring PetClinic system.

## Spring PetClinic - Web Application - Components

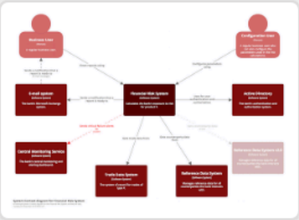



The Components diagram for the Spring PetClinic web application.


Spring PetClinic - Web Application - Components
Clinic Employee
Uses
Uses
Uses
Uses
Uses
Web Application
PetController
CrashController
VetController
VisitController
OwnerController
ClinicService
PetRepository
VetRepository
VisitRepository
OwnerRepository
Reads from and writes to
Reads from and writes t
Reads from and writes to
Reads from and writes to
Relational Database

# Structurizr

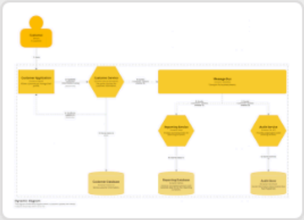
Visualise, document and explore your software architecture



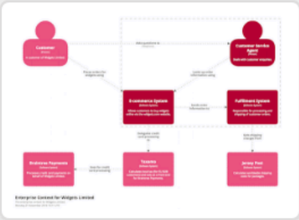
Create system context diagrams using Java code.

Extract components from your Java codebase, using static analysis and reflection.

Client-side encrypt your workspace for additional security ('password').



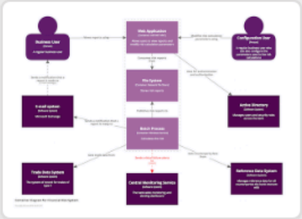
Create animated diagrams to describe dynamic behaviour.



Create enterprise context diagrams using code.

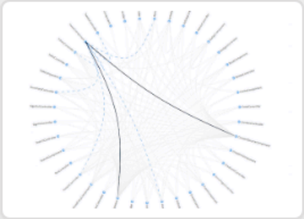
Document your software with Markdown and AsciiDoc, effortlessly embed diagrams.

Create diagrams using text with Structurizr Express.

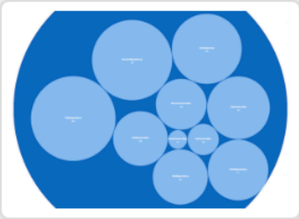


Create diagrams using C# code.

Extract components from your .NET codebase, using static analysis and reflection.



Explore inbound, outbound and cyclic dependencies between components.

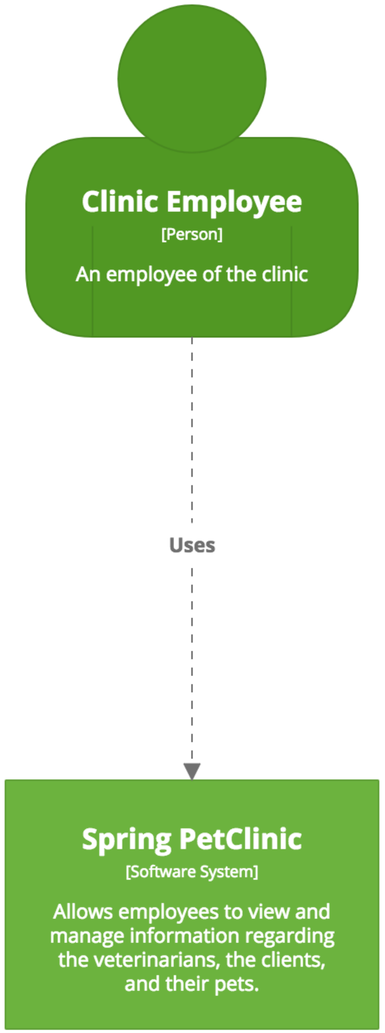


Explore component size.

Embed diagrams into Atlassian Confluence with prebuilt macros.

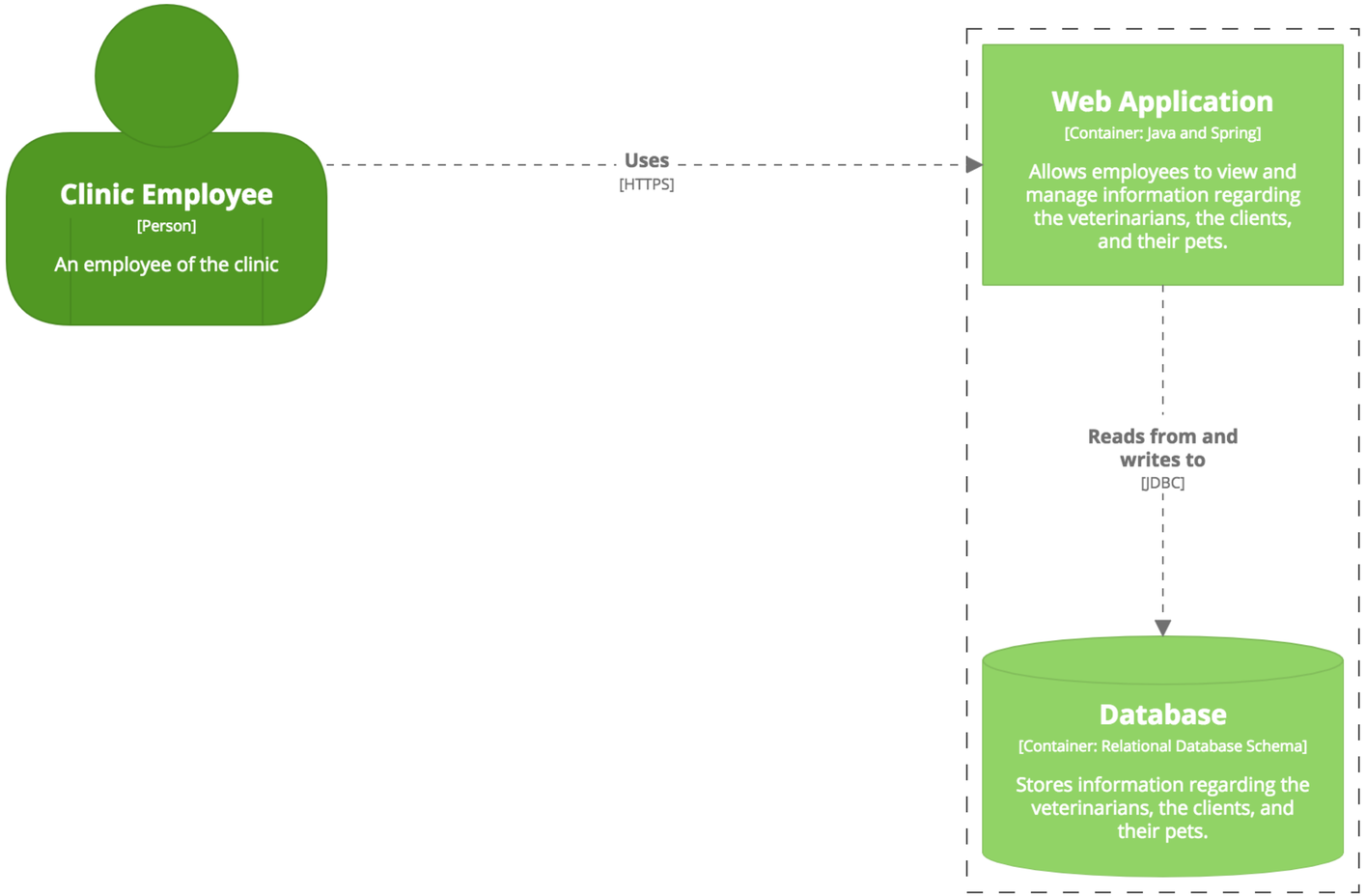
```
// upload the software architecture model to structurizr.com
StructurizrClient client = new StructurizrClient("key", "secret");
client.putWorkspace(1234, workspace);
```


```
{
 "id" : 0,
 "name" : "Spring PetClinic",
 "description" : "This is a C4 representation of the Spring PetClinic sample app (https://github.com/spring-projects/spring-petclinic/)",
 "model" : {
   "people" : [ {
     "tags" : "Element,Person",
     "id" : "2",
     "name" : "Clinic Employee",
     "description" : "An employee of the clinic",
     "relationships" : [ {
       "tags" : "Relationship,Synchronous",
       "id" : "3",
       "sourceId" : "2",
       "destinationId" : "1",
       "description" : "Uses",
       "interactionStyle" : "Synchronous"
     }, {
       "tags" : "Relationship,Synchronous",
       "id" : "6",
       "sourceId" : "2",
       "destinationId" : "4",
       "description" : "Uses",
       "technology" : "HTTP",
       "interactionStyle" : "Synchronous"
     }, {
       "tags" : "Relationship,Synchronous",
       "id" : "28",
       "sourceId" : "2",
       "destinationId" : "8",
       "description" : "Uses",
       "technology" : "HTTP",
       "interactionStyle" : "Synchronous"
```



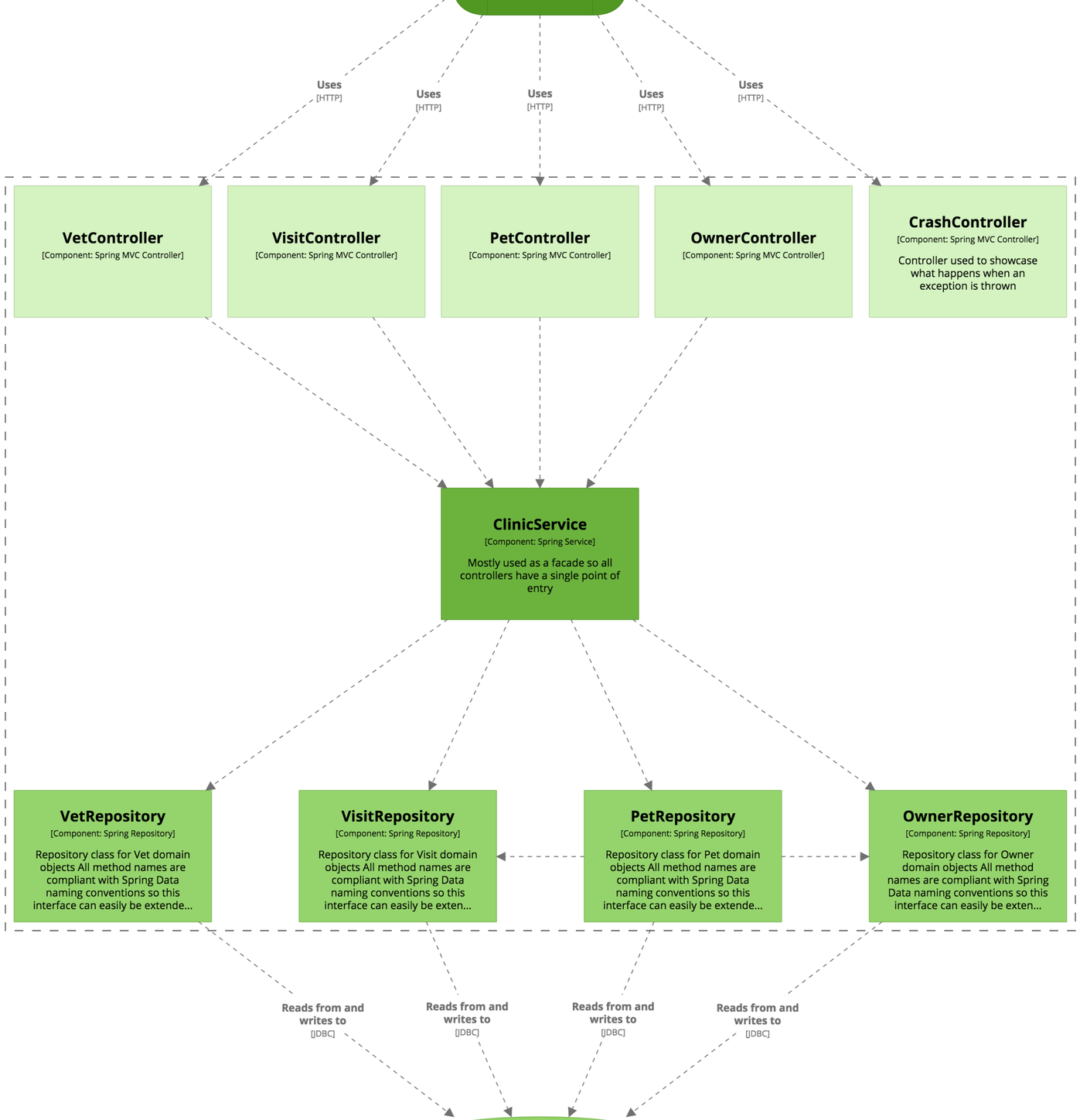

## System Context diagram for Spring PetClinic

The System Context diagram for the Spring PetClinic system.
Last modified: Monday 26 June 2017 12:59 UTC | Version: 95de1d9f8bf63560915331664b27a4a75ce1f1f6

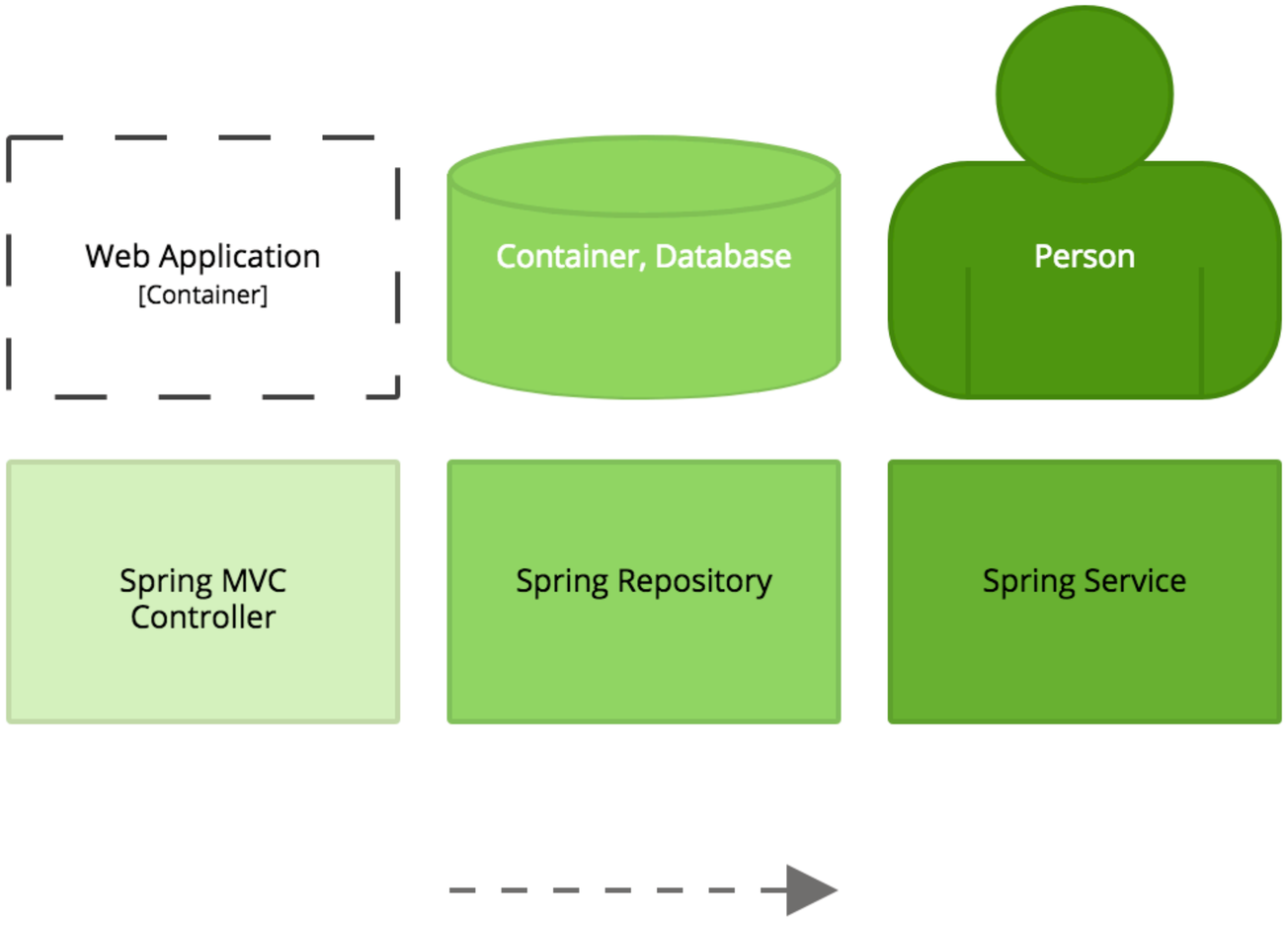


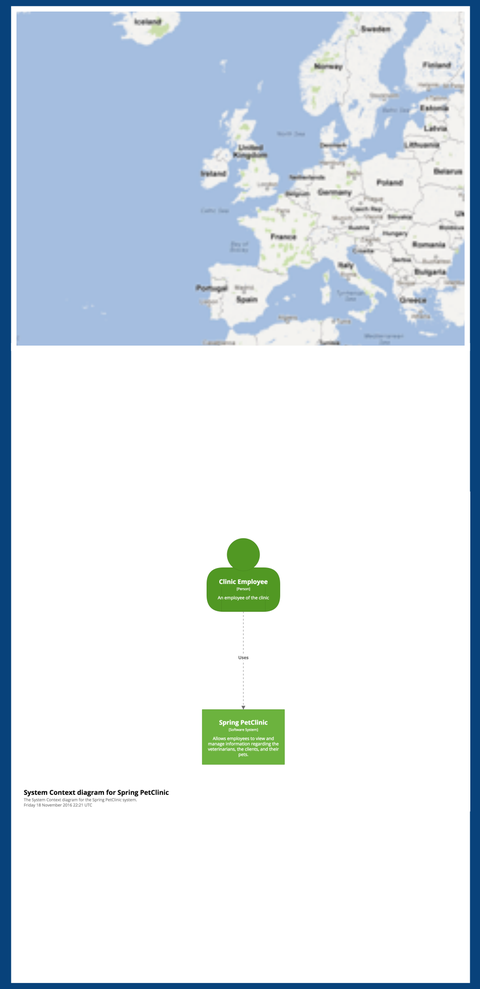

## Container diagram for Spring PetClinic

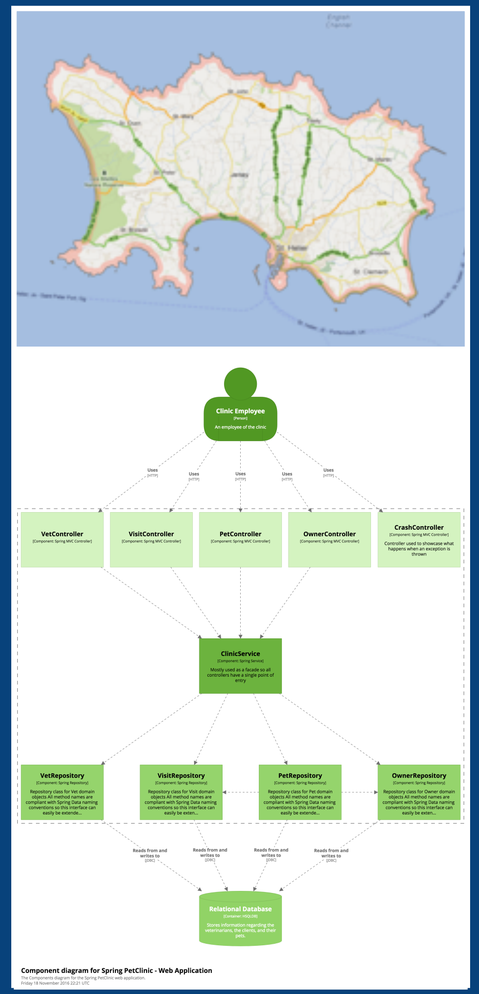
The Containers diagram for the Spring PetClinic system.
Last modified: Monday 26 June 2017 12:59 UTC | Version: 95de1d9f8bf63560915331664b27a4a75ce1f1f6

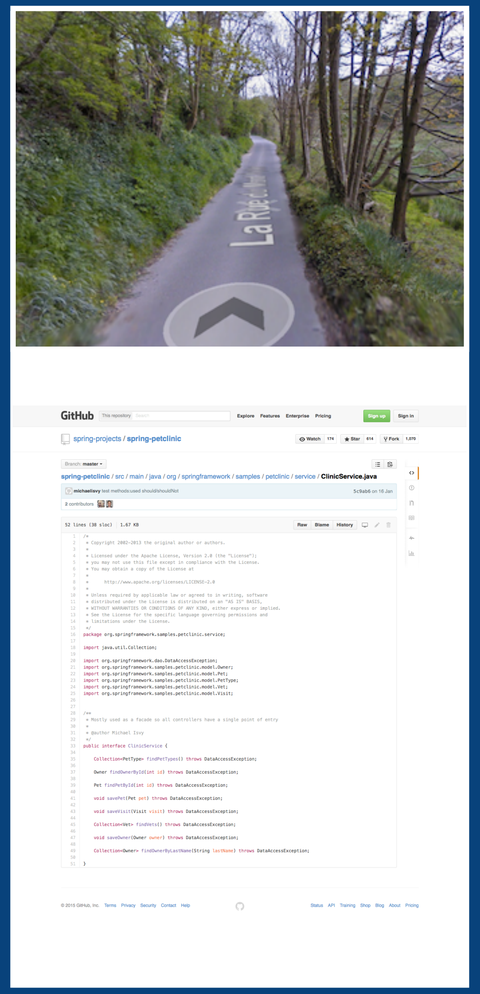

Uses
[HTTP]
Uses
[HTTP]
Uses
[HTTP]
Uses
[HTTP]
Uses
[HTTP]
VetController
[Component: Spring MVC Controller]
VisitController
[Component: Spring MVC Controller]
PetController
[Component: Spring MVC Controller]
OwnerController
[Component: Spring MVC Controller]
CrashController
[Component: Spring MVC Controller]
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
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 extende...
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 exten...
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 extende...
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 exten...
Reads from and writes to
[JDBC]
Reads from and writes to
[JDBC]
Reads from and writes to
[JDBC]
Reads from and writes to
[JDBC]

# Diagram key


Web Application
[Container]
Container, Database
Person
Spring MVC
Controller
Spring Repository
Spring Service
Relationship





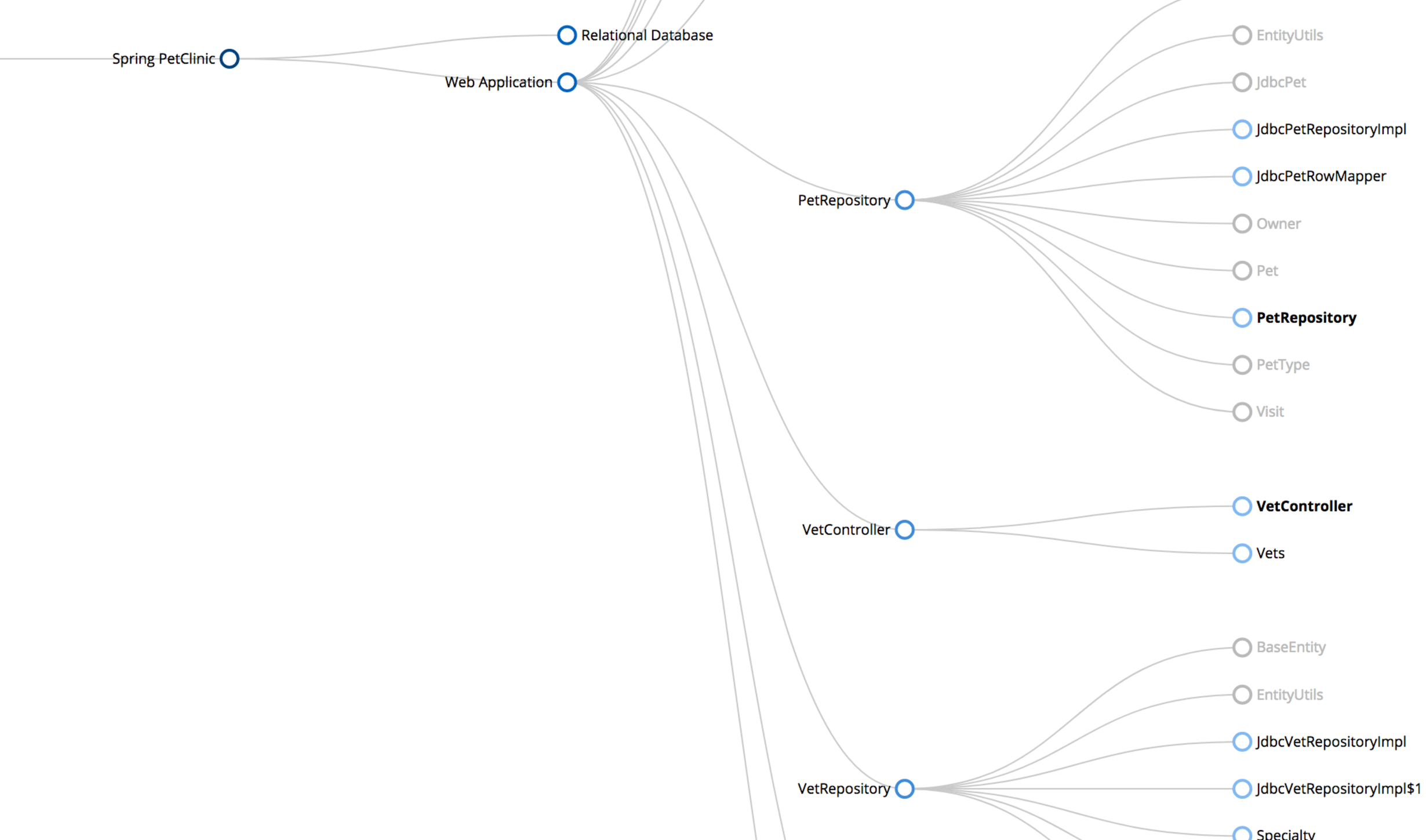



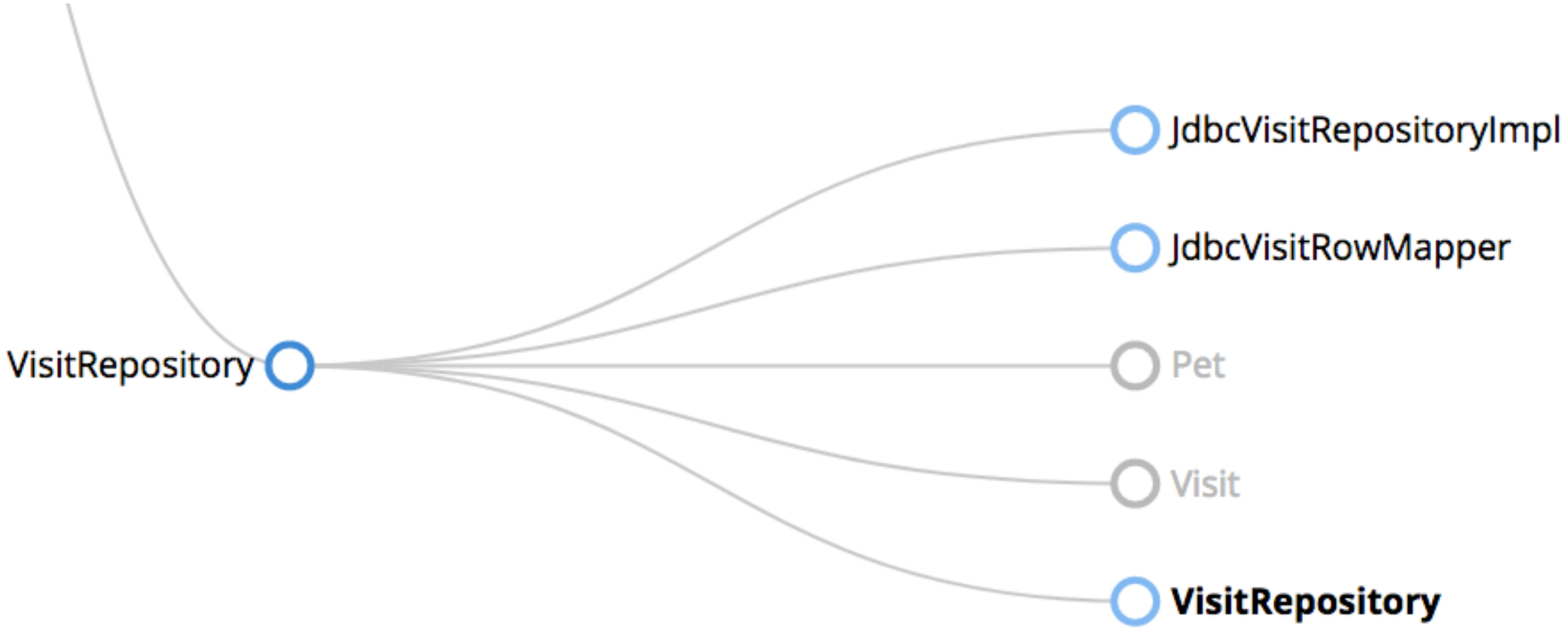

# Diagrams are maps

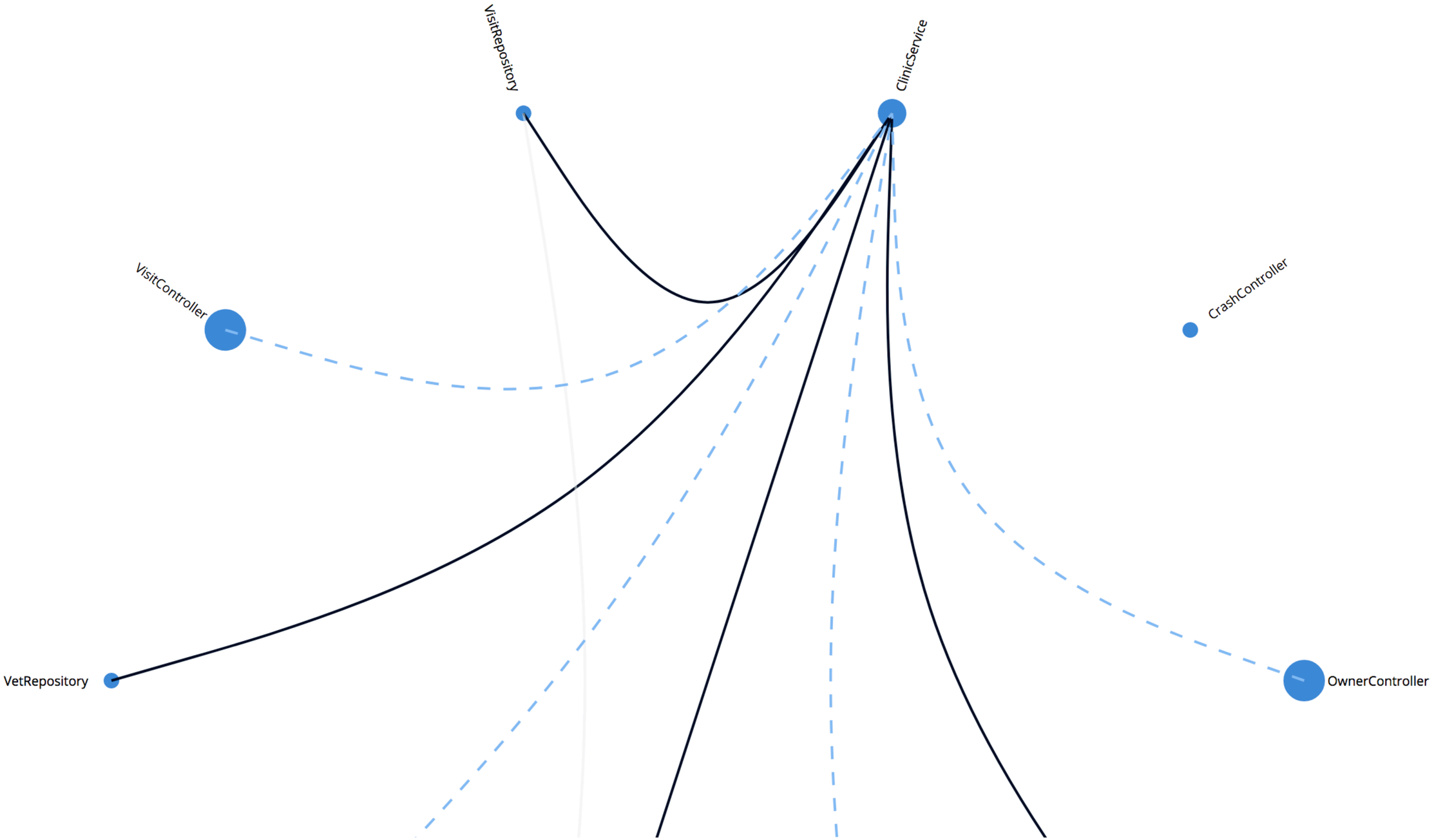
that help software developers navigate a large and/or complex codebase

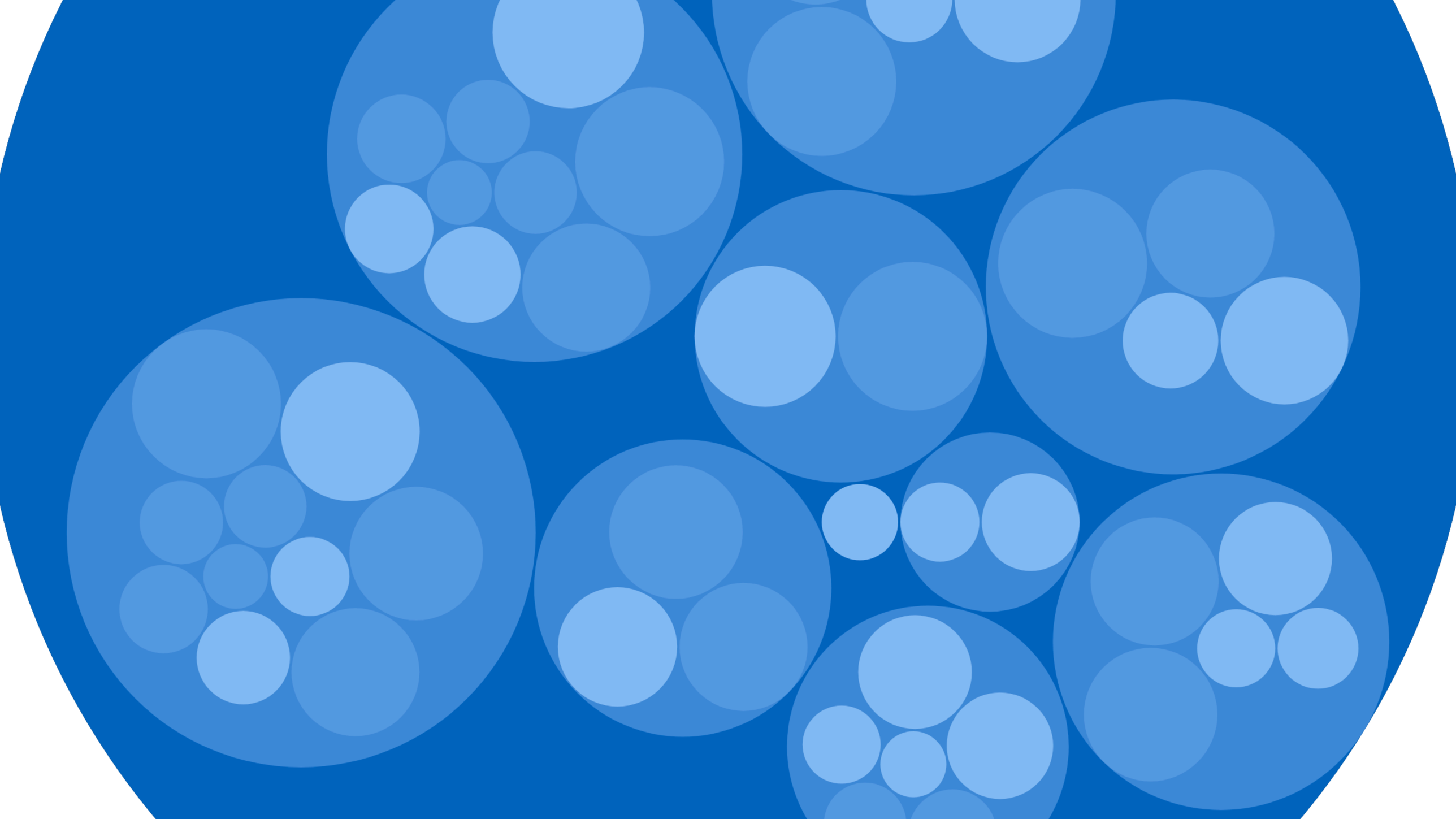
Integration with your build process keeps models up to date

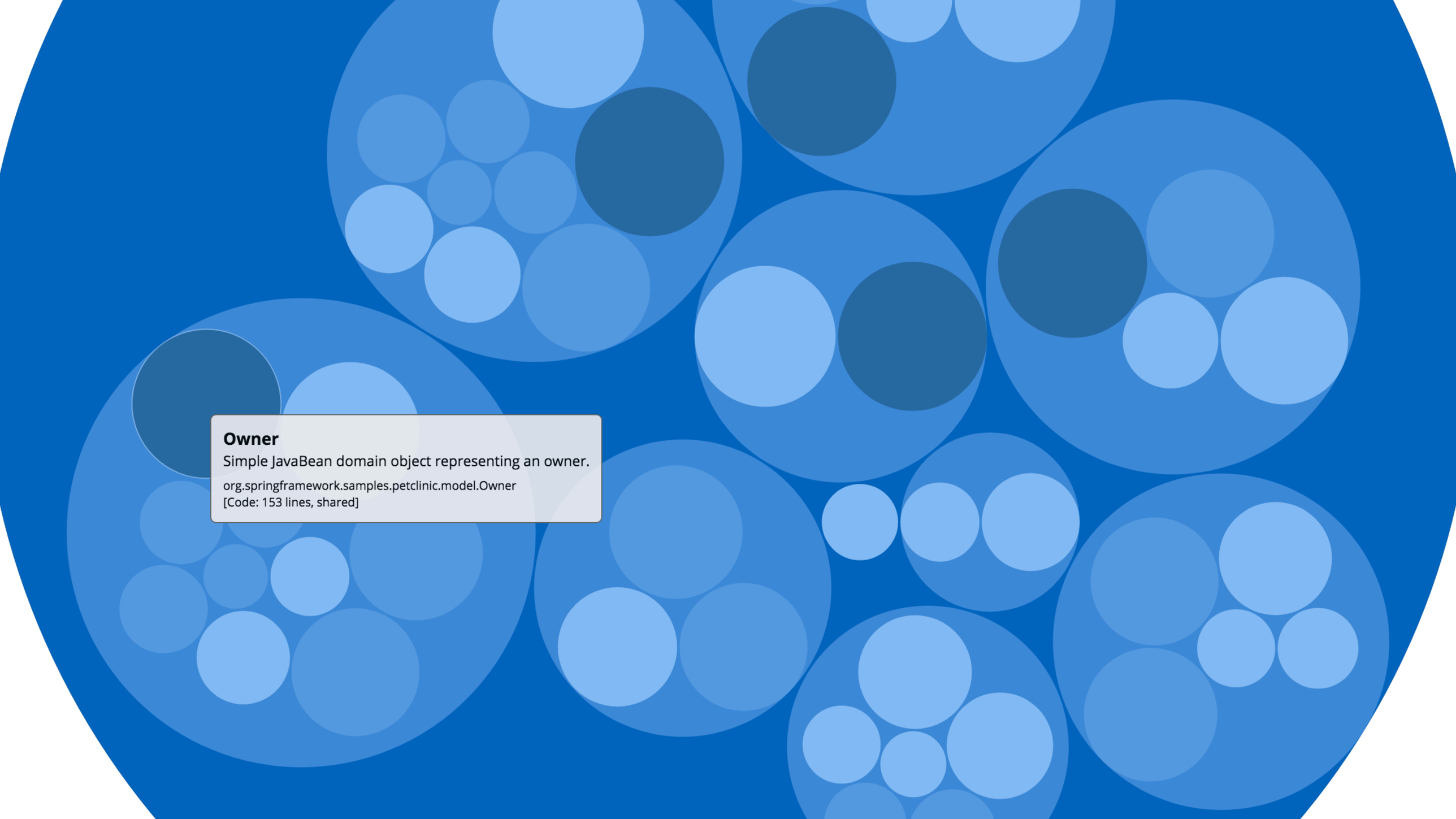
Once you have a model of your software system, you can **explore** it


Spring PetClinic
Relational Database
Web Application
PetRepository
EntityUtils
JdbcPet
JdbcPetRepositoryImpl
JdbcPetRowMapper
Owner
Pet
PetRepository
PetType
Visit
VetController
VetController
Vets
VetRepository
BaseEntity
EntityUtils
JdbcVetRepositoryImpl
JdbcVetRepositoryImpl$1
Specialty


VisitRepository
JdbcVisitRepositoryImpl
JdbcVisitRowMapper
Pet
Visit
VisitRepository


VisitRepository
ClinicService
VisitController
CrashController
VetRepository
OwnerController





**Owner**

Simple JavaBean domain object representing an owner.

org.springframework.samples.petclinic.model.Owner
[Code: 153 lines, shared]

# The **1990**’s called and they want their tools back!

It’s 2018 and we shouldn’t be using a general purpose diagramming tool for software architecture

# Abstractions first, notation second

Ensure that your team has a ubiquitous language to describe software architecture

# Thank you!

simon.brown@codingthearchitecture.com
@simonbrown