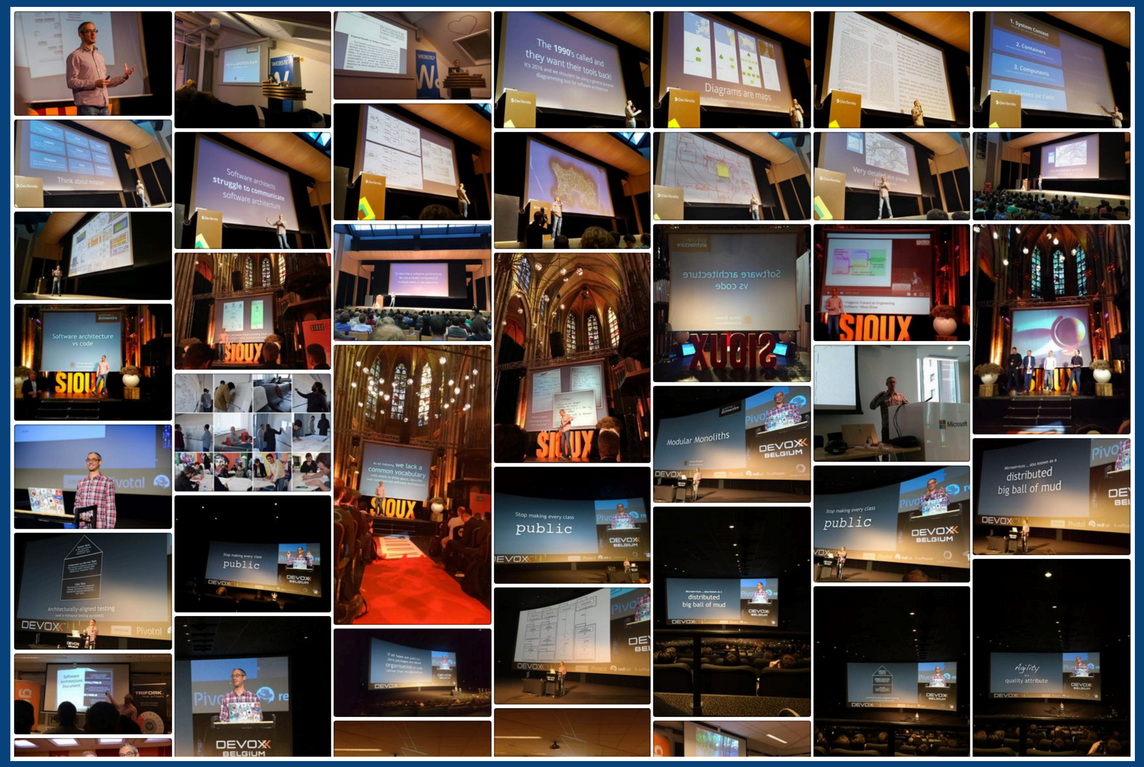
Software documentation

Why, what and how?

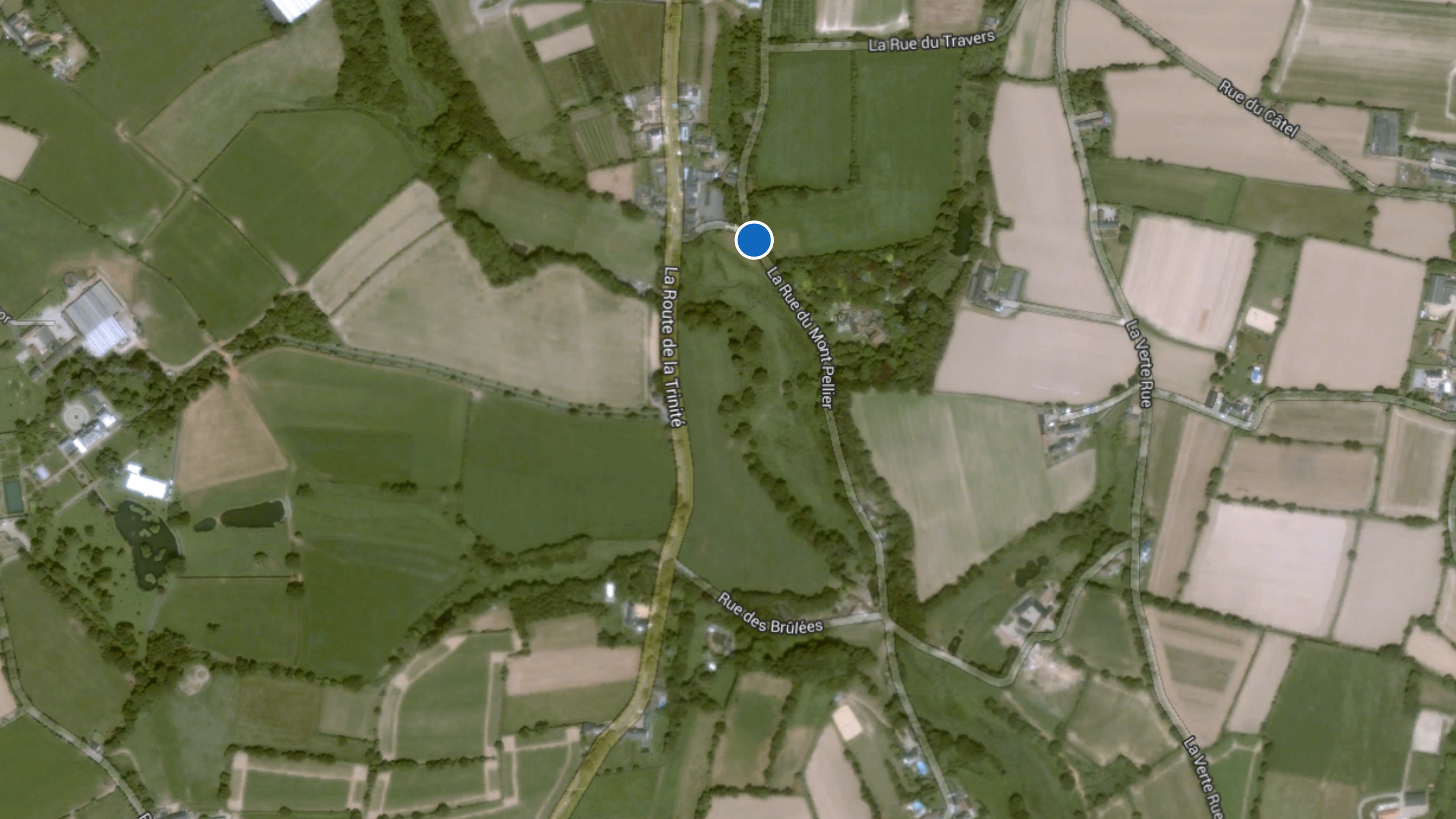
@simonbrown

Simon Brown

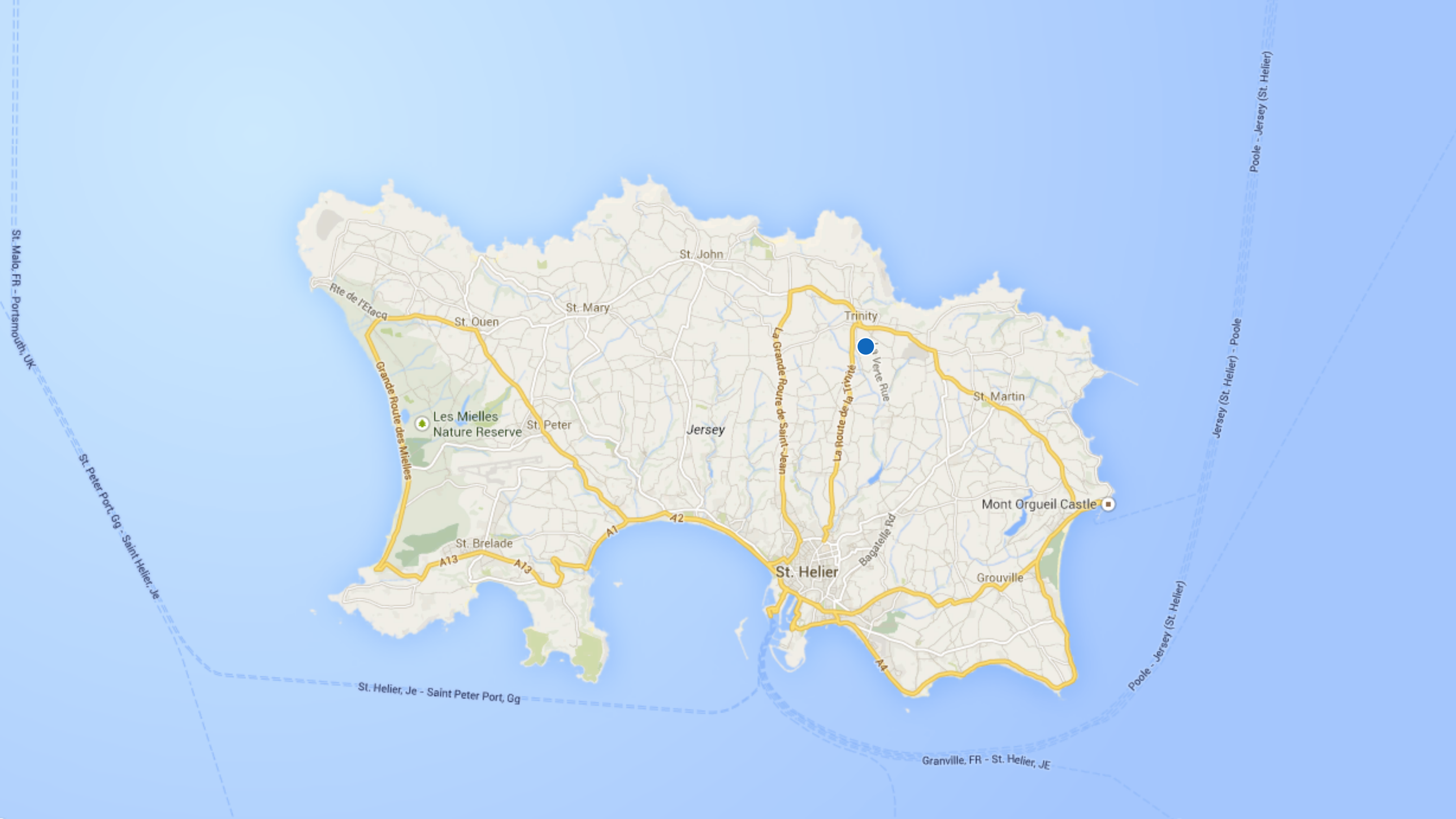
I'm an independent software development consultant specialising in software architecture, and the author of “Software Architecture for Developers”; a developer-friendly guide to software architecture, technical leadership and the balance with agility. I regularly speak at software development conferences, meetups and organisations around the world; delivering keynotes, presentations, training courses and workshops.

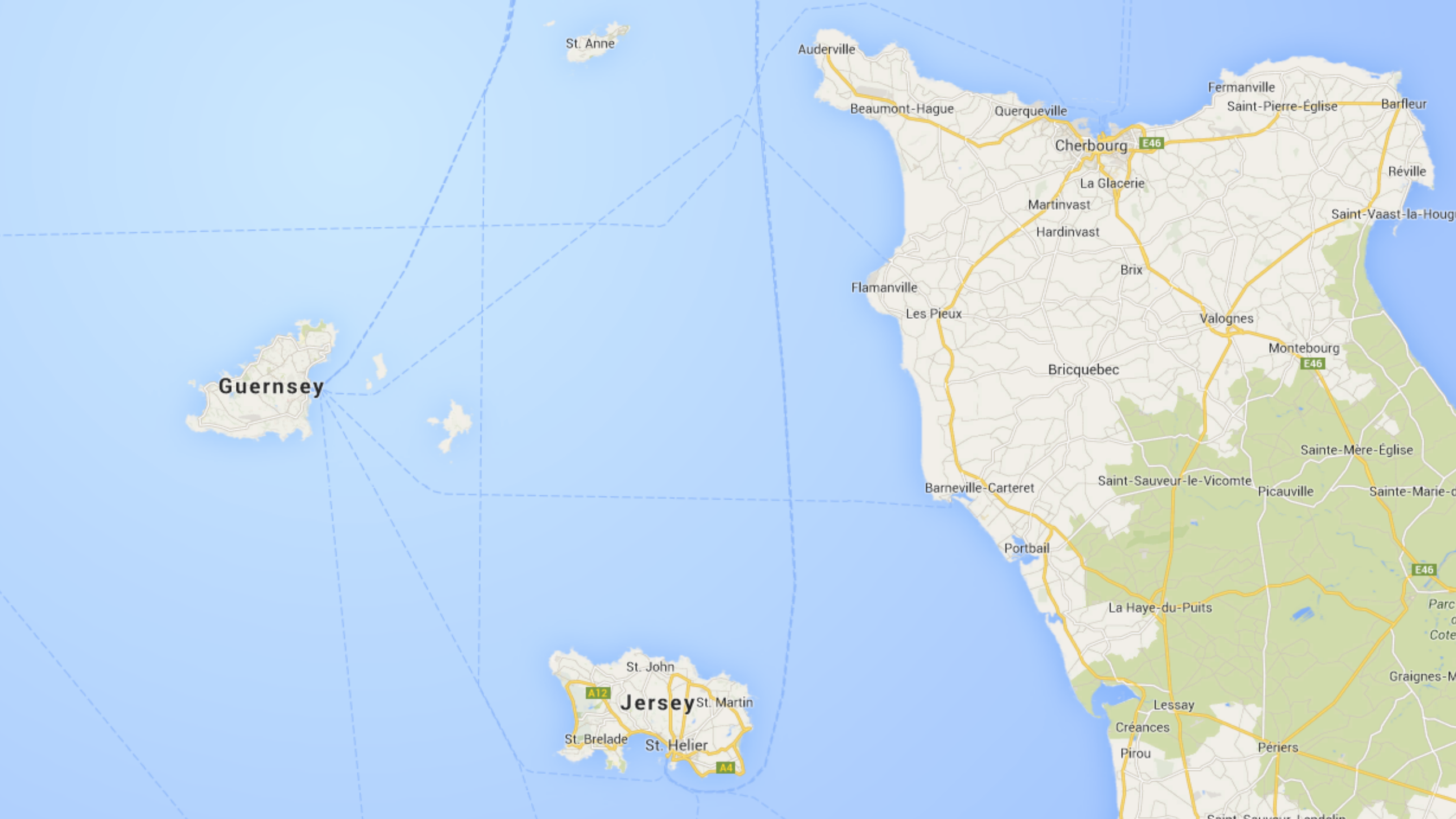


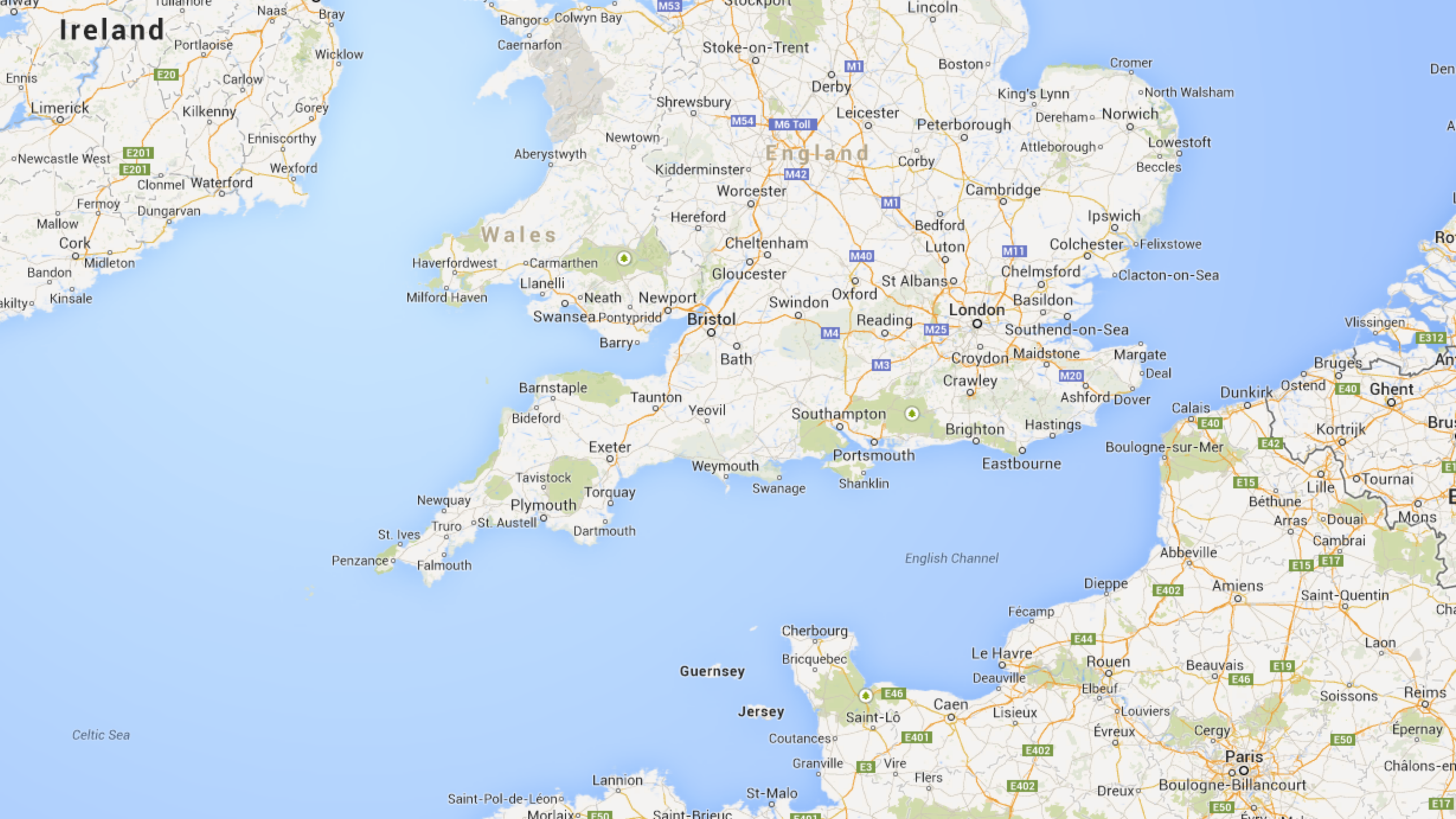

La Rue du Mont


La Rue du Travers
Rue du Câtel
La Route de la Trinité
La Rue du Mont Pellier
La Verte Rue
Rue des Brûlées

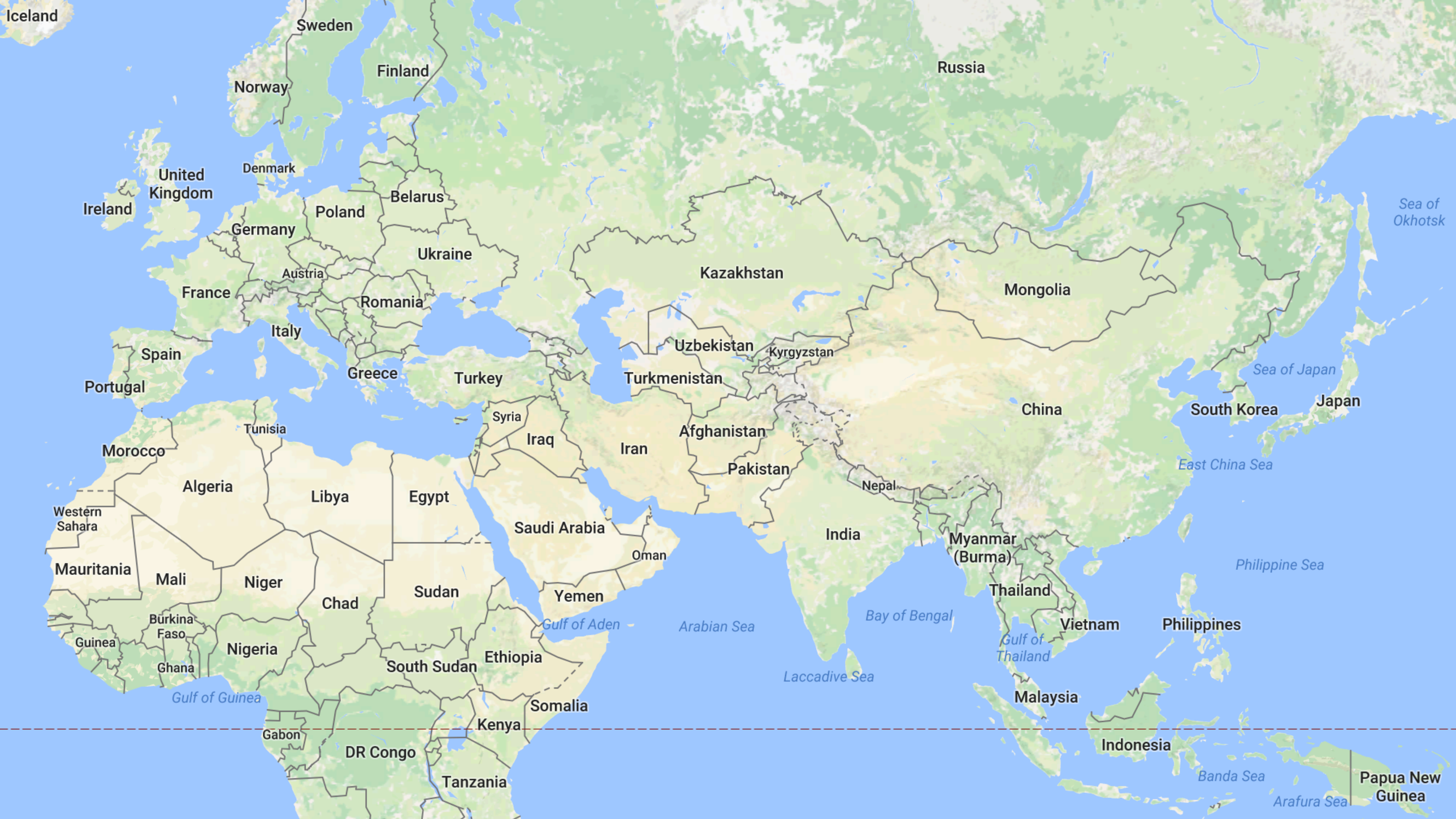

St. John
St. Mary
St. Ouen
Trinity
St. Martin
St. Peter
Jersey
Les Mielles
Nature Reserve
Mont Orgueil Castle
St. Brelade
St. Helier
Grouville
St. Helier, Je - Saint Peter Port, Gg
Granville, FR - St. Helier, JE
St. Malo, FR - Portsmouth, UK
St. Peter Port, Gg - Saint Helier, Je
Poole - Jersey (St. Helier)
Jersey (St. Helier) - Poole


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
A1
A2
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)


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

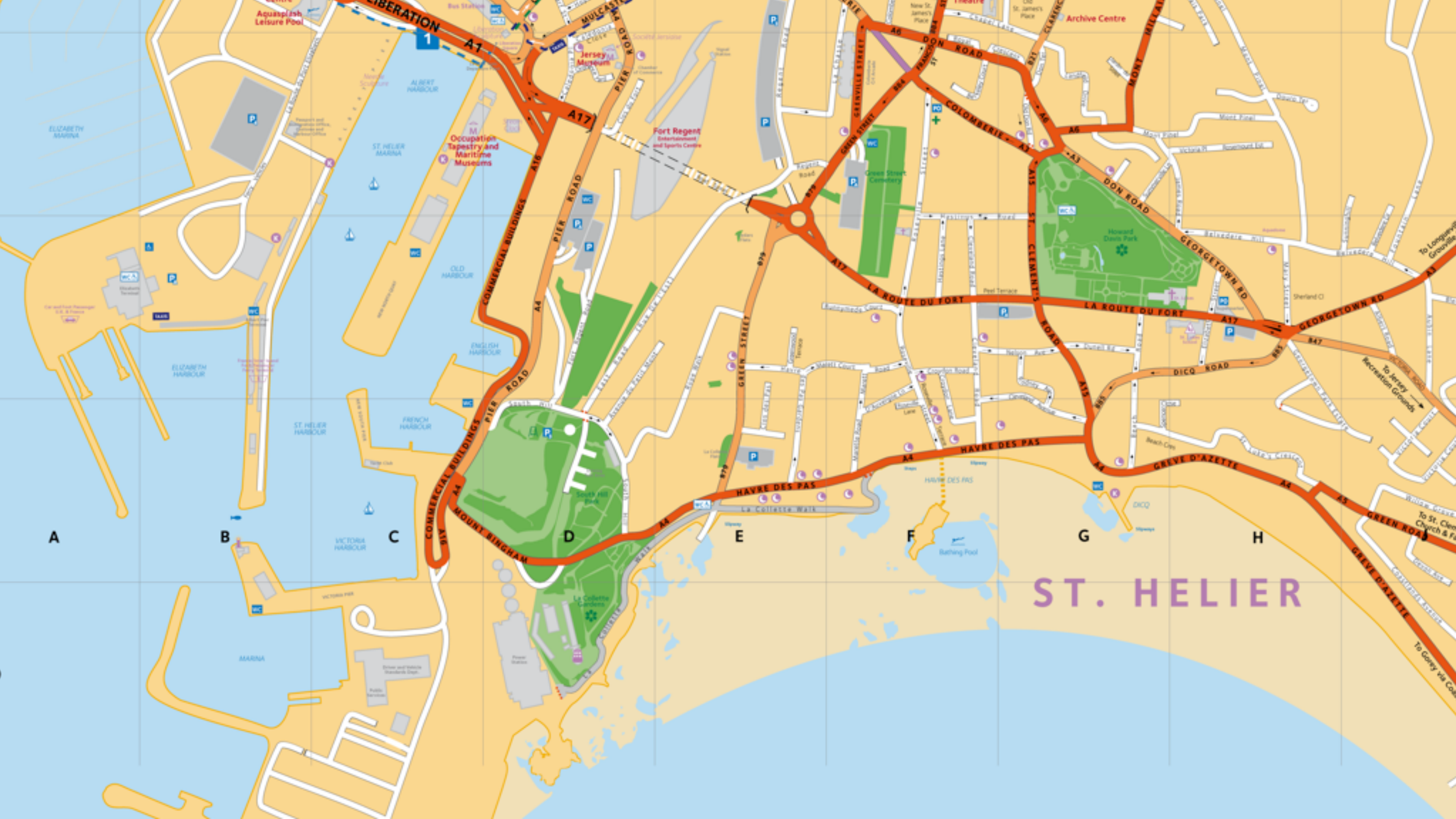

Ireland
England
Wales
Celtic Sea
English Channel
Guernsey
Jersey
Dublin
Belfast
Cork
Limerick
Waterford
Galway
Wexford
Kilkenny
Carlow
Portlaoise
Wicklow
Bray
Naas
Ennis
Clonmel
Dungarvan
Fermoy
Mallow
Midleton
Bandon
Kinsale
Newcastle West
Enniscorthy
Gorey
Tullamore
Bangor
Colwyn Bay
Caernarfon
Aberystwyth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

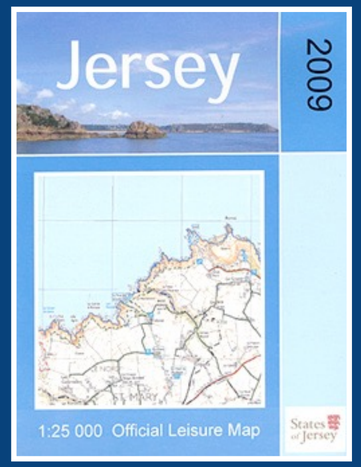

North Sea
Viborg
Aarhus
Silkeborg
Horsens
Helsingborg
Kristianstad
Karlskrona
Copenhagen
Malmö
Esbjerg
Denmark
Odense
Elsinore
Kiel
Heide
Stralsund
Rostock
Swinoujscie
Anklam
Gryfice
Schwerin
Neubrandenburg
Szczecin
Stargard Szczecinski
Bremerhaven
Hamburg
Bergedorf
Aurich
Groningen
Bremen
Den Helder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

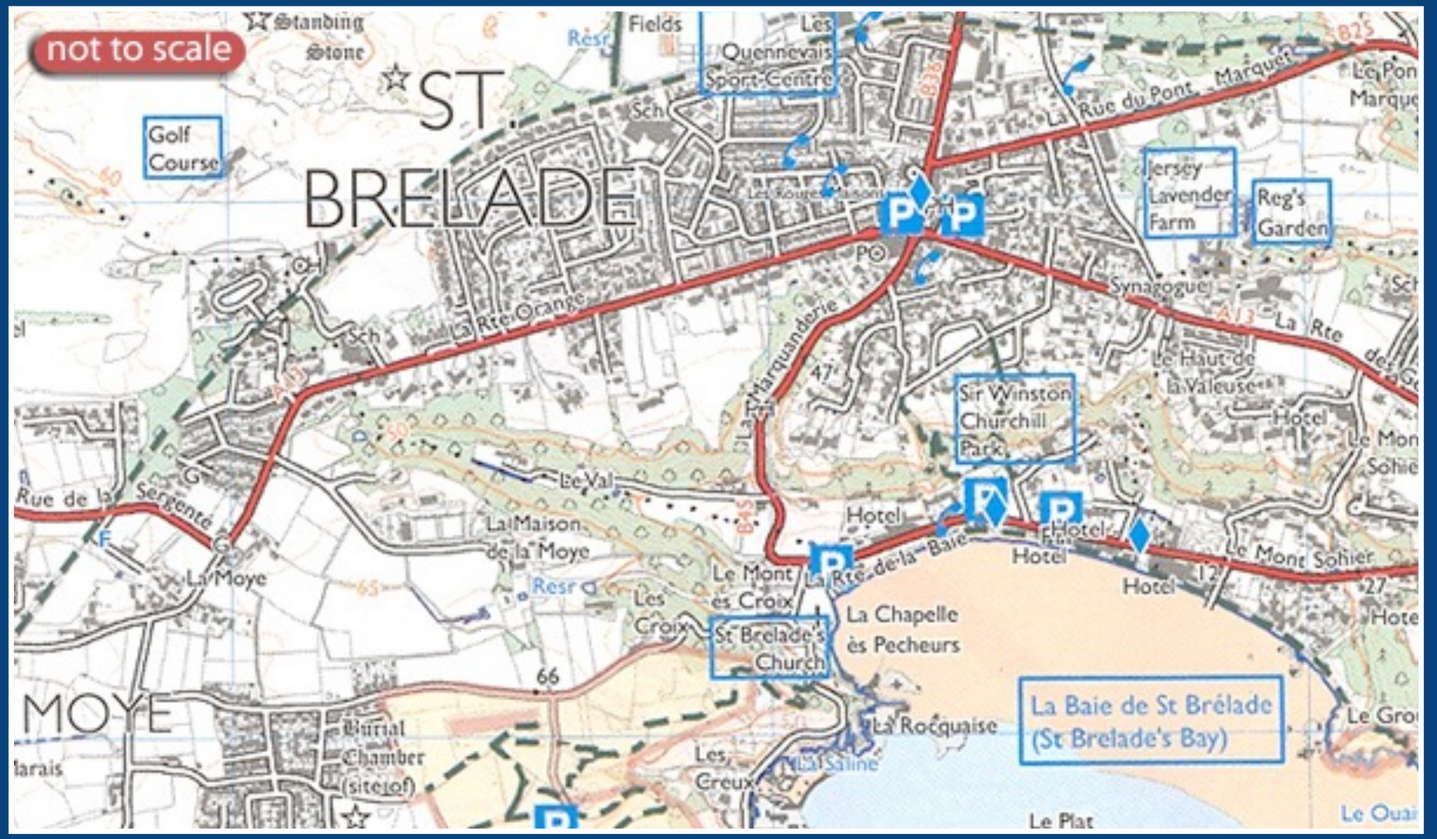

Iceland
Sweden
Norway
Finland
Russia
United Kingdom
Ireland
Denmark
Belarus
Poland
Germany
Ukraine
Austria
France
Romania
Italy
Spain
Portugal
Greece
Turkey
Kazakhstan
Mongolia
Uzbekistan
Kyrgyzstan
Turkmenistan
Sea of Okhotsk
Sea of Japan
China
South Korea
Japan
Syria
Iraq
Iran
Afghanistan
Pakistan
Nepal
India
East China Sea
Tunisia
Morocco
Algeria
Libya
Egypt
Western Sahara
Saudi Arabia
Oman
Myanmar (Burma)
Philippine Sea
Mauritania
Mali
Niger
Chad
Sudan
Yemen
Thailand
Gulf of Aden
Arabian Sea
Bay of Bengal
Vietnam
Philippines
Burkina Faso
Guinea
Nigeria
Ghana
Gulf of Thailand
Ethiopia
South Sudan
Laccadive Sea
Malaysia
Gulf of Guinea
Somalia
Kenya
Gabon
Indonesia
DR Congo
Banda Sea
Papua New Guinea
Tanzania
Arafura Sea

Enough detail to start **exploring**


01
02
03
Grosnez
La Grève au Lançon (Plémont)
Plémont
Grosnez Castle (ruins)
Creux Gros
Creux Gabourel
Douet de la Mer
Race Course
Les Landes
Portinfer
C105
B55
Rue de la Pointe
Ville la Bas
Les Pallières
B56
Route de Vinchelez
Le Pinacle
Model Aircraft Field
B34
La Route de Millais
Le Pulec
GREVE DE LECQ
Grève de Lecq Barracks
Creux de Lasse
Les Reuses
Devil's Hole
Rifle Range
Crabbé
Bleu Soleil Campsite
B65
Léoville
B40
B55
La Villaise
B35
Sea Wall
L'Étacq
Treasures of the Earth
Battle of Flowers Museum
C114
C122
C115
B64
Route du Marais
Les Prés
Jersey Pearl
Channel Islands Military Museum
La Route des Laveurs
Barge Aground
La Grande Route des Mielles
1
La Mielle de Morville
Dolmen du Mont Grantez
Le Canné
La Crabière
ST. OUEN
Ville au Bas
St Ouen's Parish Church
C117
La Ville de L'Eglise
St Ouen's Manor
A12
La Grande Route de St. Ouen
La Grande Route de St. Pierre
aMaizin! Maze Adventure Park & Craft Centre
B69
B32
B26
C106
Val de la Mare Reservoir
St. Ouen's Pond
Les Mielles
Le Mont Rossignol
6
Mont du Jubilé
B41
St Peter's Parish Church & Hall
ST. OUEN'S
Sorel
La Houle
Ronez
Le Fosse Vicq
Go Kart Racing
Moto Cross
Mourier Bay
ST. JOHN'S BAY
Wolf's Caves
Frémont
C100
C99
Devil's Hole
Mourier Valley
Les Landes
ST. JOHN
C101
St John's Parish Church
B52
B63
La Rue de L'Etocquet
La Mare Wine Estate
C104
B33
ST. MARY
Mont Mado
B67
A9
A10
B50
Route des Issues
Le Claire Stables
La Route de St. Jean
C103
La Rue du Pont
St Mary's Parish Hall
B26
B39
4
Jersey Goldsmiths & Lion Park
La Grande Route de St. Laurent
Bon Air
Handois Reservoir
B27
BSJA Showground
Carrefour
La Rue de la Vallée
The Elms
C112
6a
Living Legend
Le Moulin de Quétivel
Ville Emphrie


ST. HELIER
A
B
C
D
E
F
G
H
LIBERATION
A1
A17
A16
A4
A6
A3
A15
B79
B84
B47
B85
PIER ROAD
COMMERCIAL BUILDINGS
MULCASTER STREET
GREEN STREET
GRENVILLE STREET
DON ROAD
COLOMBERIE
MONT MILLAIS
LA ROUTE DU FORT
ST. CLEMENT'S ROAD
GEORGETOWN RD
DICQ ROAD
HAVRE DES PAS
MOUNT BINGHAM
GREVE D'AZETTE
GREEN ROAD
Aquasplash Leisure Pool
Jersey Museum
Fort Regent
Entertainment and Sports Centre
Occupation Tapestry and Maritime Museums
Archive Centre
Howard Davis Park
Green Street Cemetery
South Hill Park
La Collette Gardens
La Collette Walk
Bathing Pool
Peel Terrace
Beach Cres
To Jersey Recreation Grounds
ALBERT HARBOUR
ELIZABETH MARINA
ST. HELIER MARINA
OLD HARBOUR
ENGLISH HARBOUR
ELIZABETH HARBOUR
ST. HELIER HARBOUR
FRENCH HARBOUR
VICTORIA HARBOUR
MARINA





Very detailed and precise

(terrain, buildings, etc)

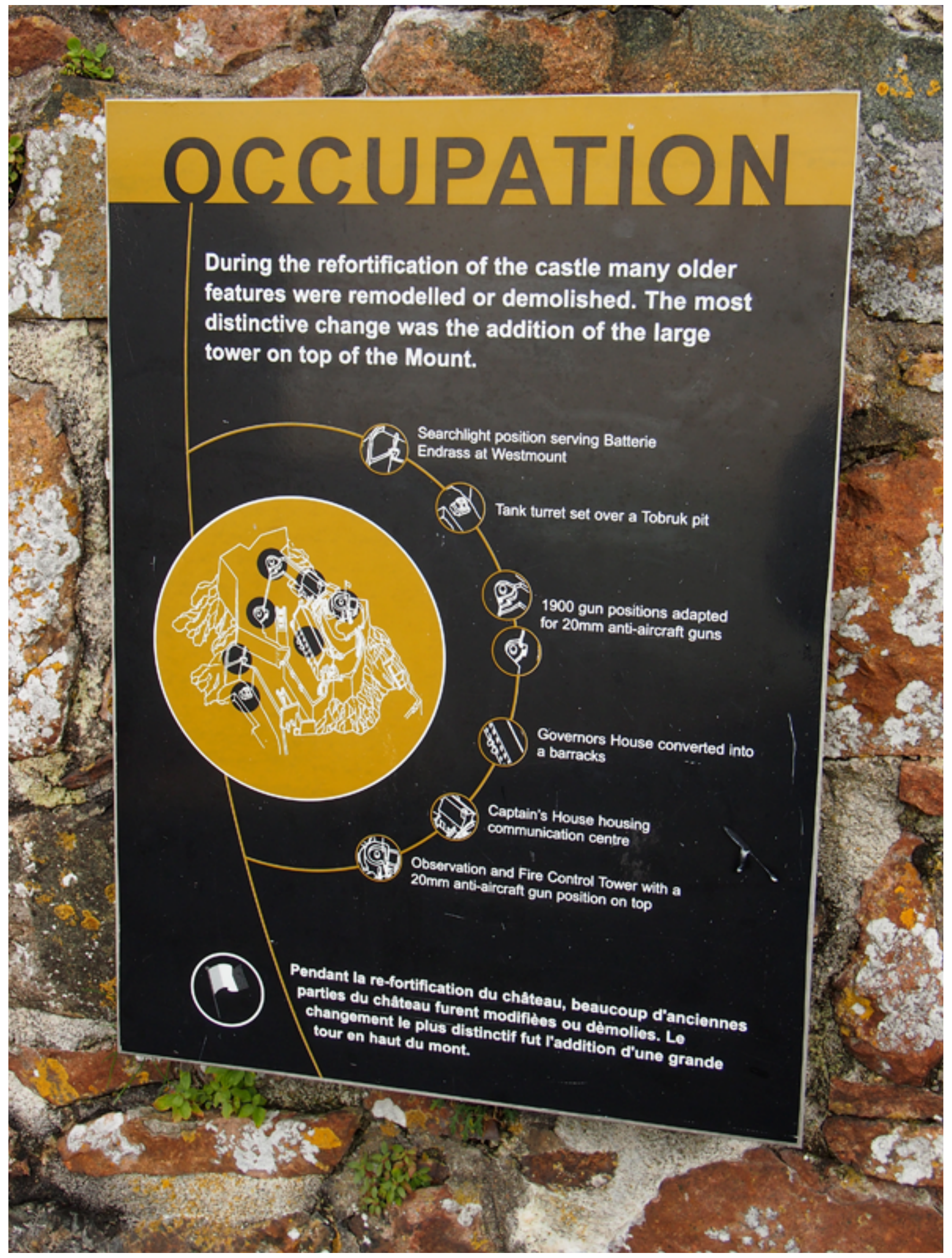
Points of interest

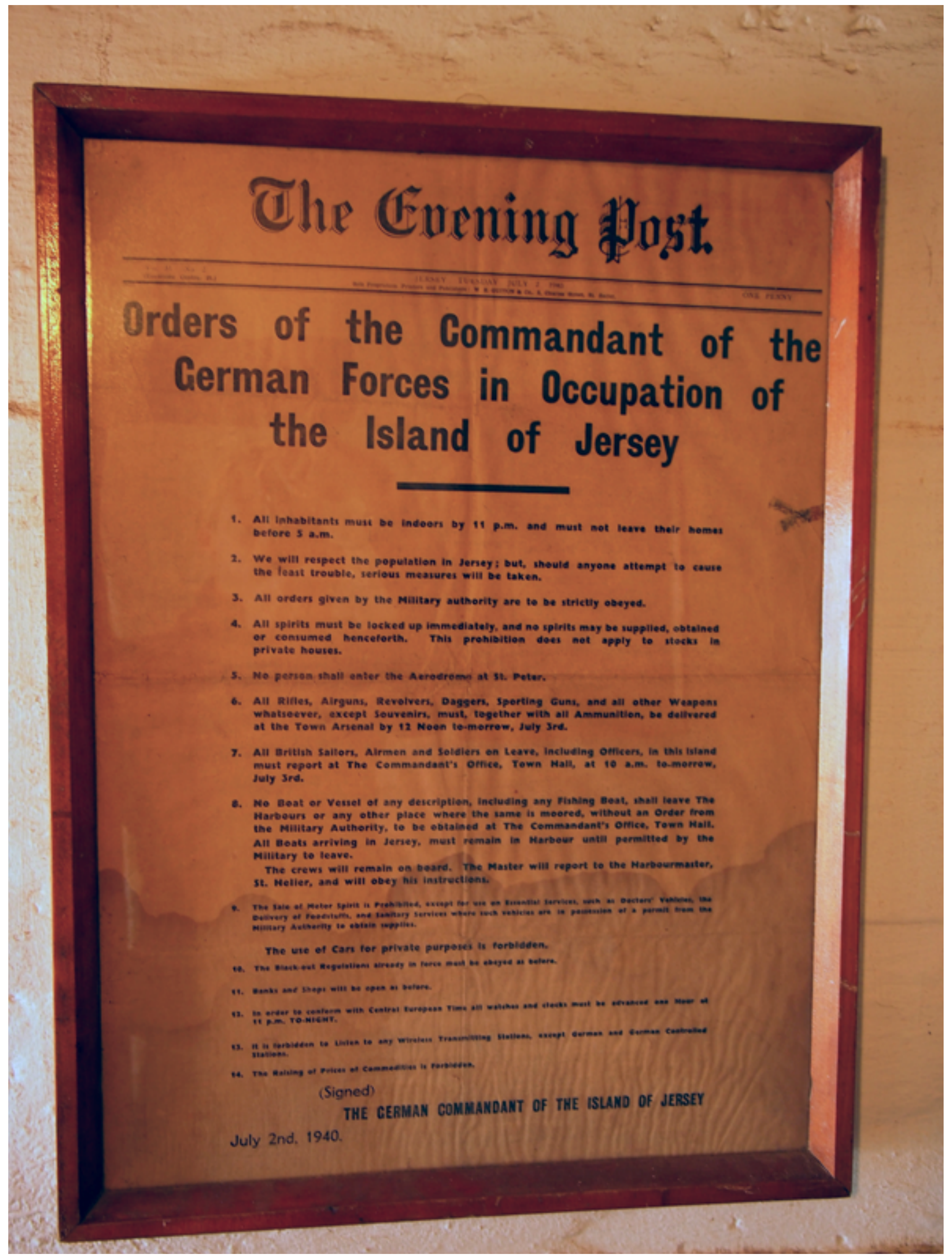
Elizabeth Castle



Granite and concrete?!

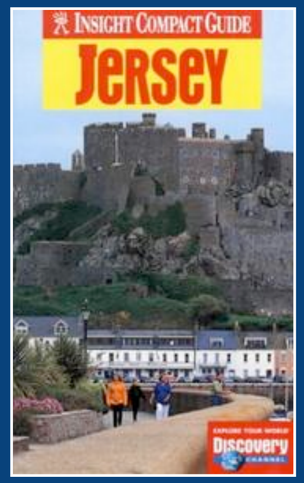

Magazine
1594
The powder and munitions
store for the original castle.
The entrance doorway was
built c1680. In 1617 the
Royal Commissioners' inventory
of the castle recorded
41 barrels of powder stored
here as well as quantities of
Dépôt des munitions.
En 1617 l'inventaire
du château fait mention de

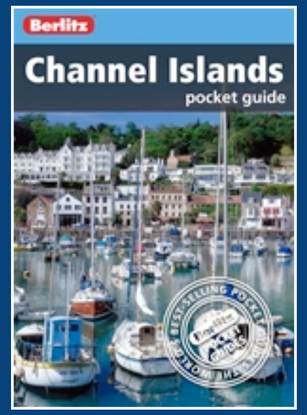

OCCUPATION
During the refortification of the castle many older features were remodelled or demolished. The most distinctive change was the addition of the large tower on top of the Mount.
Searchlight position serving Batterie Endrass at Westmount
Tank turret set over a Tobruk pit
1900 gun positions adapted for 20mm anti-aircraft guns
Governors House converted into a barracks
Captain's House housing communication centre
Observation and Fire Control Tower with a 20mm anti-aircraft gun position on top
Pendant la re-fortification du château, beaucoup d'anciennes parties du château furent modifiées ou démolies. Le changement le plus distinctif fut l'addition d'une grande tour en haut du mont.

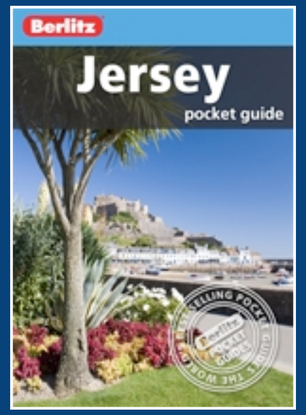

The Evening Post
Orders of the Commandant of the German Forces in Occupation of the Island of Jersey
1. All inhabitants must be indoors by 11 p.m. and must not leave their homes before 5 a.m.
2. We will respect the population in Jersey; but, should anyone attempt to cause the least trouble, serious measures will be taken.
3. All orders given by the Military authority are to be strictly obeyed.
4. All spirits must be locked up immediately, and no spirits may be supplied, obtained or consumed henceforth. This prohibition does not apply to stocks in private houses.
5. No person shall enter the Aerodrome at St. Peter.
6. All Rifles, Airguns, Revolvers, Daggers, Sporting Guns, and all other Weapons whatsoever, except Souvenirs, must, together with all Ammunition, be delivered at the Town Arsenal by 12 Noon to-morrow, July 3rd.
7. All British Sailors, Airmen and Soldiers on Leave, including Officers, in this Island must report at The Commandant's Office, Town Hall, at 10 a.m. to-morrow, July 3rd.
8. No Boat or Vessel of any description, including any Fishing Boat, shall leave The Harbours or any other place where the same is moored, without an Order from the Military Authority, to be obtained at The Commandant's Office, Town Hall. All Boats arriving in Jersey, must remain in Harbour until permitted by the Military to leave.
The crews will remain on board. The Master will report to the Harbourmaster, St. Helier, and will obey his instructions.
9. The Sale of Motor Spirit is Prohibited, except for use on Essential Services, such as Doctors' Vehicles, the Delivery of Foodstuffs, and Sanitary Services where such vehicles are in possession of a permit from the Military Authority to obtain supplies.
The use of Cars for private purposes is forbidden.
10. The Black-out Regulations already in force must be obeyed as before.
11. Banks and Shops will be open as before.
12. In order to conform with Central European Time all watches and clocks must be advanced one Hour at 11 p.m. TO-NIGHT.
13. It is forbidden to Listen to any Wireless Transmitting Stations, except German and German Controlled Stations.
14. The Raising of Prices of Commodities is Forbidden.
(Signed)
THE GERMAN COMMANDANT OF THE ISLAND OF JERSEY
July 2nd. 1940.

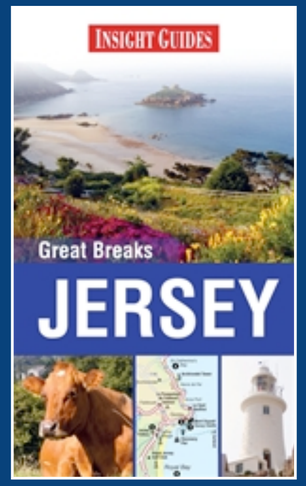
History


OCCUPATION









Travel Guidebook

(maps, points of interest, sights, itineraries, history, culture, practical information, etc)

Working software

over

comprehensive documentation

Manifesto for Agile Software Development

What is your current strategy for documenting software?

What are your goals for this workshop?

Audience

Identify the different audiences for software documentation

1. Who are they?
2. What do they want to know?

Rank the audiences you’ve identified according to how technical they are

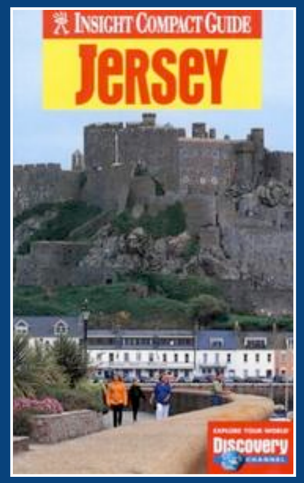
When might you need to deliver documentation?

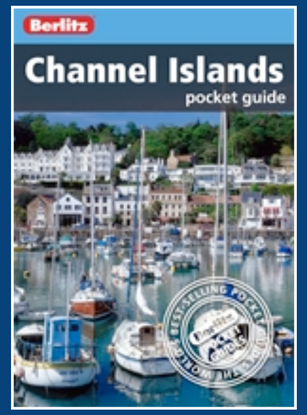
Documentation

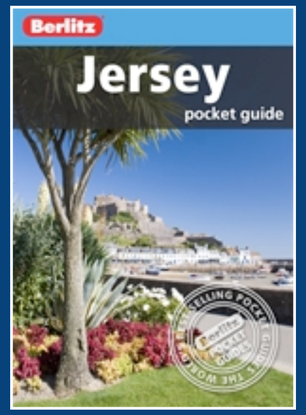
The code doesn’t tell the whole story

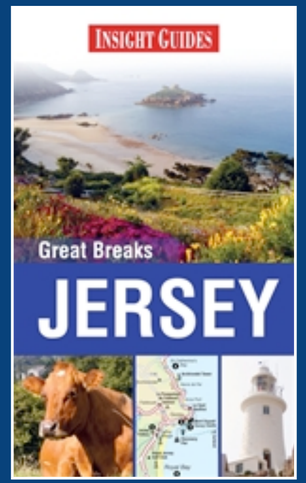
Software Architecture Document

Useful information spread across hundreds of pages; rarely read or updated









Software Guidebook

(maps, points of interest, sights, itineraries, history, culture, practical information, etc)

Describe what you

can’t get from the code

Documentation should be **constantly evolving**

The scope is a single **software system**

Context

A system context diagram, plus some narrative text to “set the scene”

Functional Overview

An overview of the software system, perhaps including wireframes, UI mockups/screenshots, workflow diagrams, business process diagrams, etc

Quality Attributes

A list of the quality attributes that
you’re aiming to satisfy

Constraints

A list of the environmental constraints you’ve identified

Principles

A list of the development and architecture principles that you want to adopt

Software Architecture

A description of the software architecture,
including static structure and dynamic behaviour

Code

A description of component implementation details, patterns, frameworks, etc

Data

Data models, entity relationship diagrams, security, volumes, archiving strategies, backup strategies, etc

Infrastructure Architecture

A description of the infrastructure available to host your software system

Deployment

The mapping of software to infrastructure

Development Environment

A description of how a new developer gets started

Operation and Support

An overview of how the software system operated, supported, monitored, etc

Decision Log

Free format text or a collection of
“Architecture Decision Records”

Title These documents have names that are short noun phrases. For example, "ADR 1: Deployment on Ruby on Rails 3.0.10" or "ADR 9: LDAP for Multitenant Integration"

Context This section describes the forces at play, including technological, political, social, and project local. These forces are probably in tension, and should be called out as such. The language in this section is value-neutral. It is simply describing facts.

Decision This section describes our response to these forces. It is stated in full sentences, with active voice. "We will ..."

Status A decision may be "proposed" if the project stakeholders haven't agreed with it yet, or "accepted" once it is agreed. If a later ADR changes or reverses a decision, it may be marked as "deprecated" or "superseded" with a reference to its replacement.

Consequences This section describes the resulting context, after applying the decision. All consequences should be listed here, not just the "positive" ones. A particular decision may have positive, negative, and neutral consequences, but all of them affect the team and project in the future.

“Architecture Decision Record”

A short description of an architecturally significant decision

http://thinkrelevance.com/blog/2011/11/15/documenting-architecture-decisions (Michael Nygard)

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

As architects, we define
the **significant decisions**

Architecture

Programming language
Monolith, microservices or hybrid approach

Design

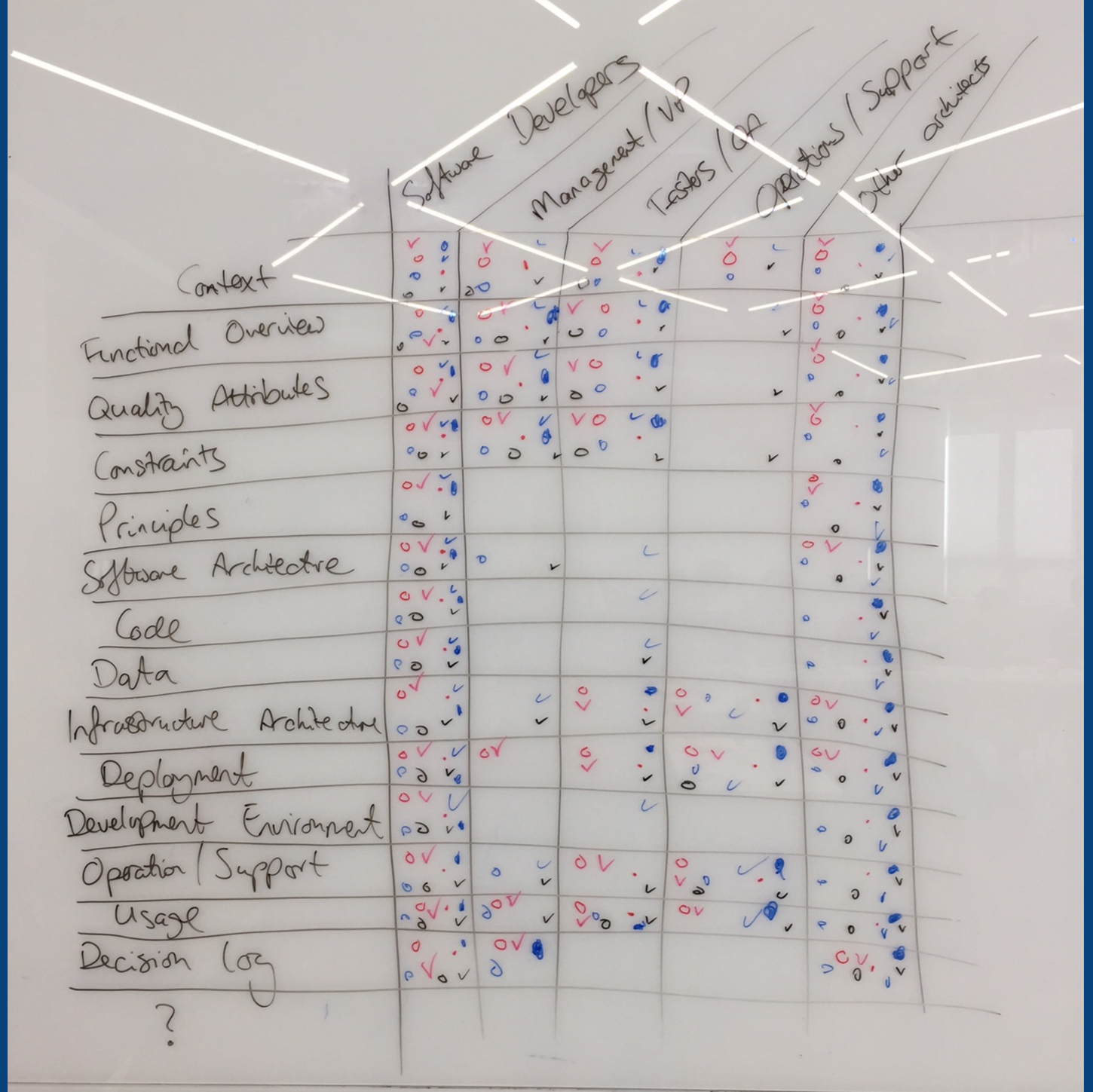
Implementation

Curly braces on the same or next line
Whitespace vs tabs

1. **Introduction and goals**: Requirements, stakeholder, (top) quality goals
2. **Constraints**: Technical and organizational constraints, conventions
3. **Context and scope**: Business and technical context, external interfaces
4. **Solution strategy**: Fundamental solution decisions and ideas
5. **Building block view**: Abstractions of source code, black-/whiteboxes
6. **Runtime view**: Runtime scenarios: How do building blocks interact
7. **Deployment view**: Hardware and technical infrastructure, deployment
8. **Crosscutting concepts**: Recurring solution approaches and patterns
9. **Architecture decisions**: Important decisions
10. **Quality**: Quality tree and quality scenarios
11. **Risks and technical debt**: Known problems, risks and technical debt
12. **Glossary**: Definitions of important business and technical terms

arc42

Which sections would you recommend each audience type to read?


Software Developers
Management / VP
Testers / QA
Operations / Support
Other architects
Context
Functional Overview
Quality Attributes
Constraints
Principles
Software Architecture
Code
Data
Infrastructure Architecture
Deployment
Development Environment
Operation / Support
Usage
Decision Log
?

How long?

Something I can read in 1-2 hours;
a good starting point for exploring the code

How do you keep software architecture documentation up to date?

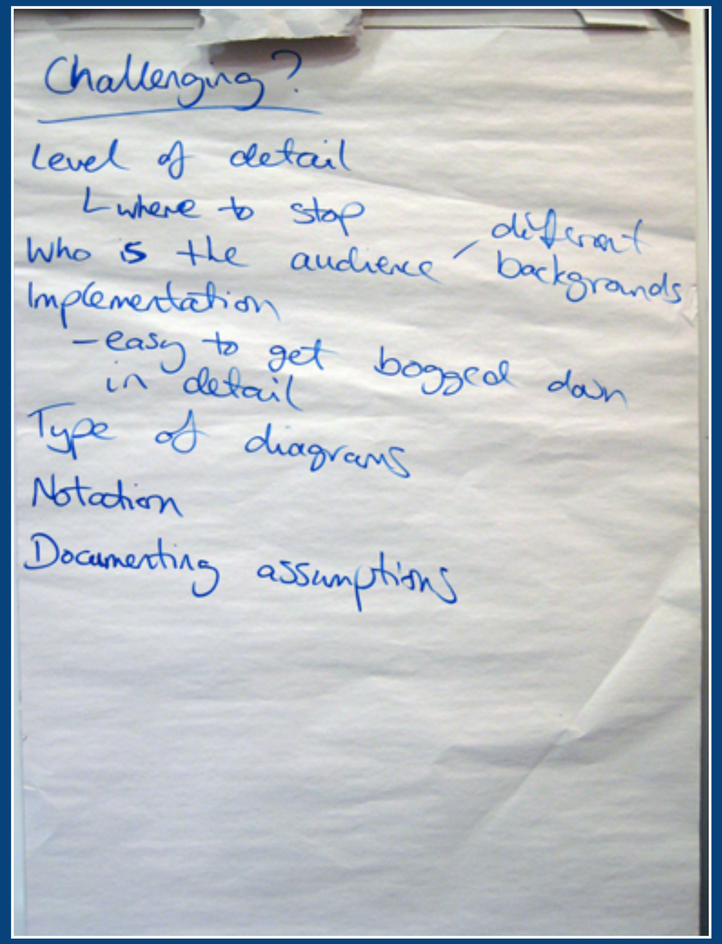
Diagrams

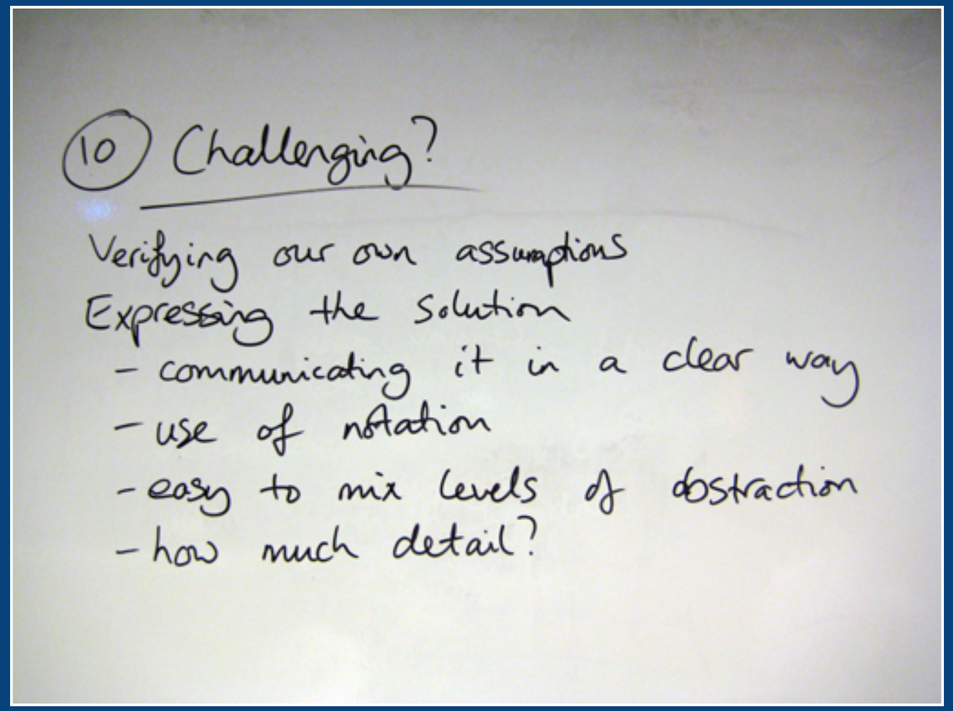
What type of diagrams could you use to help you document software?

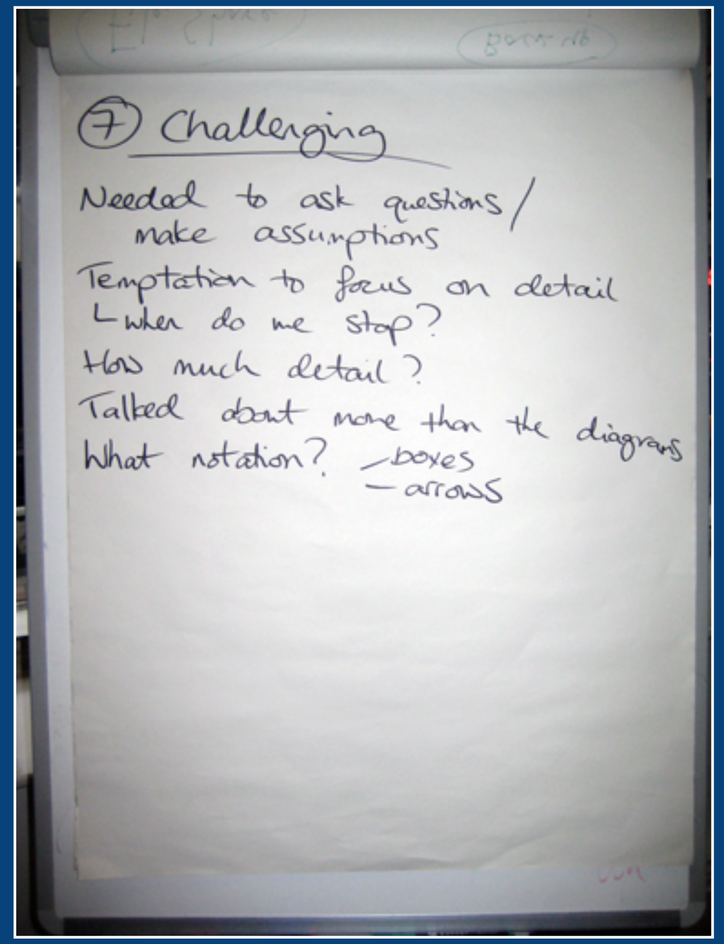
Design a software solution for the ”Financial Risk System”, and **draw** one or more architecture diagrams to describe your solution

Did you find this exercise

challenging?







Did you find this exercise challenging?

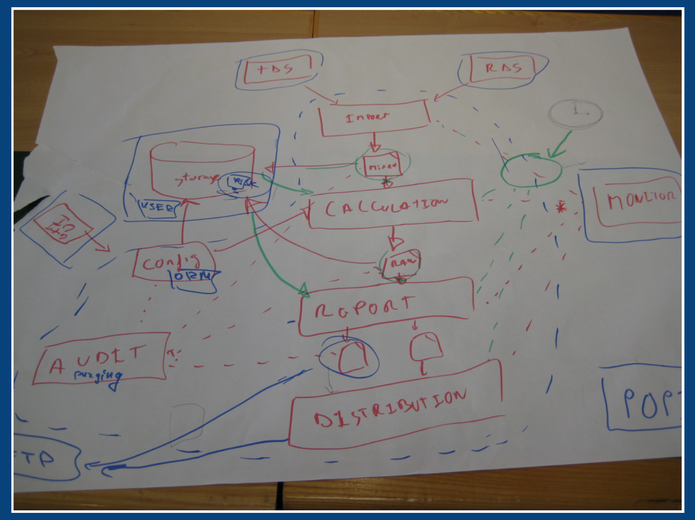
This doesn’t make sense,
but we’ll explain it.

Information is likely
still stuck in your heads

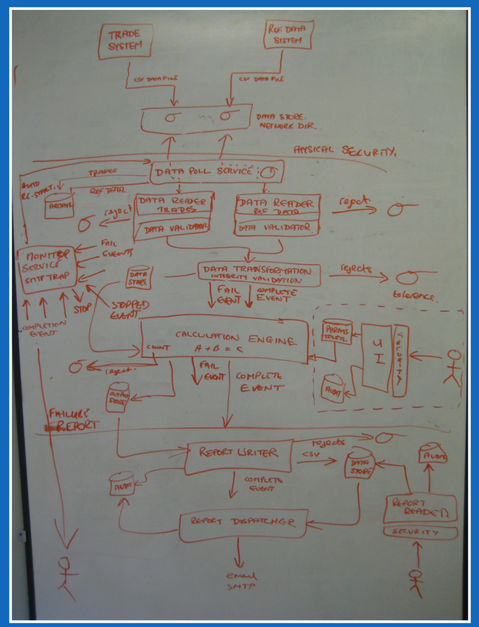
Swap and **review** your diagrams

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


1
1
1

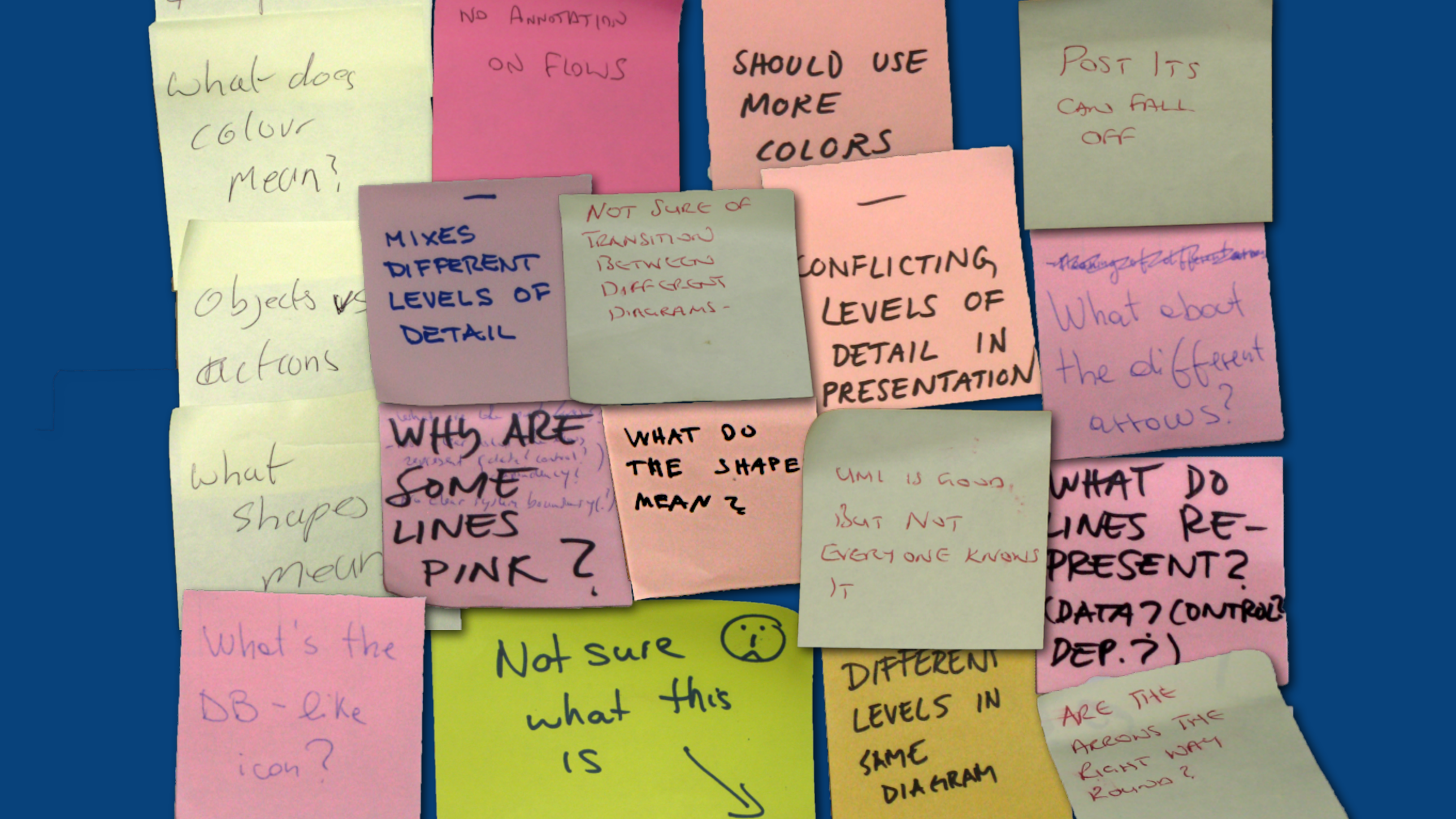

TBS
RBS
Import
storage
VSED
CALCULATION
CONFIG
OPEN
REPORT
AUDIT
purging
DISTRIBUTION


2
2
2


TRADE SYSTEM
REF DATA SYSTEM
DATA PULL SERVICE
DATA READER TRADES
DATA READER REF DATA
DATA VALIDATOR
DATA TRANSFORMATION INTEGRITY VALIDATION
CALCULATION ENGINE
A + B = C
COMPLETE EVENT
FAIL EVENT
REPORT WRITER
REPORT DISPATCHER
PHYSICAL SECURITY
MONITOR SERVICE
STOPPED EVENT
FAILURE REPORT
UI
email SMTP

Documentation and diagrams should be able to stand on their own

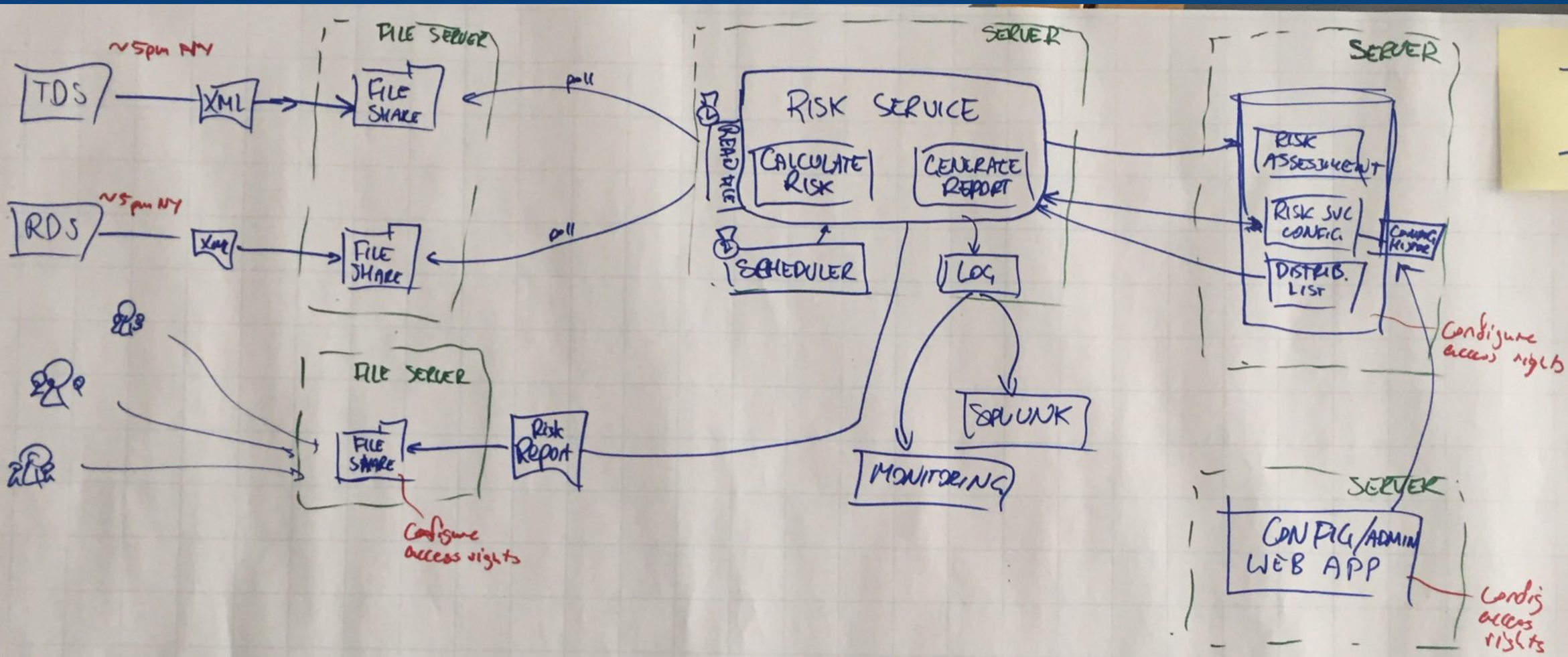
(to some extent, anyway)

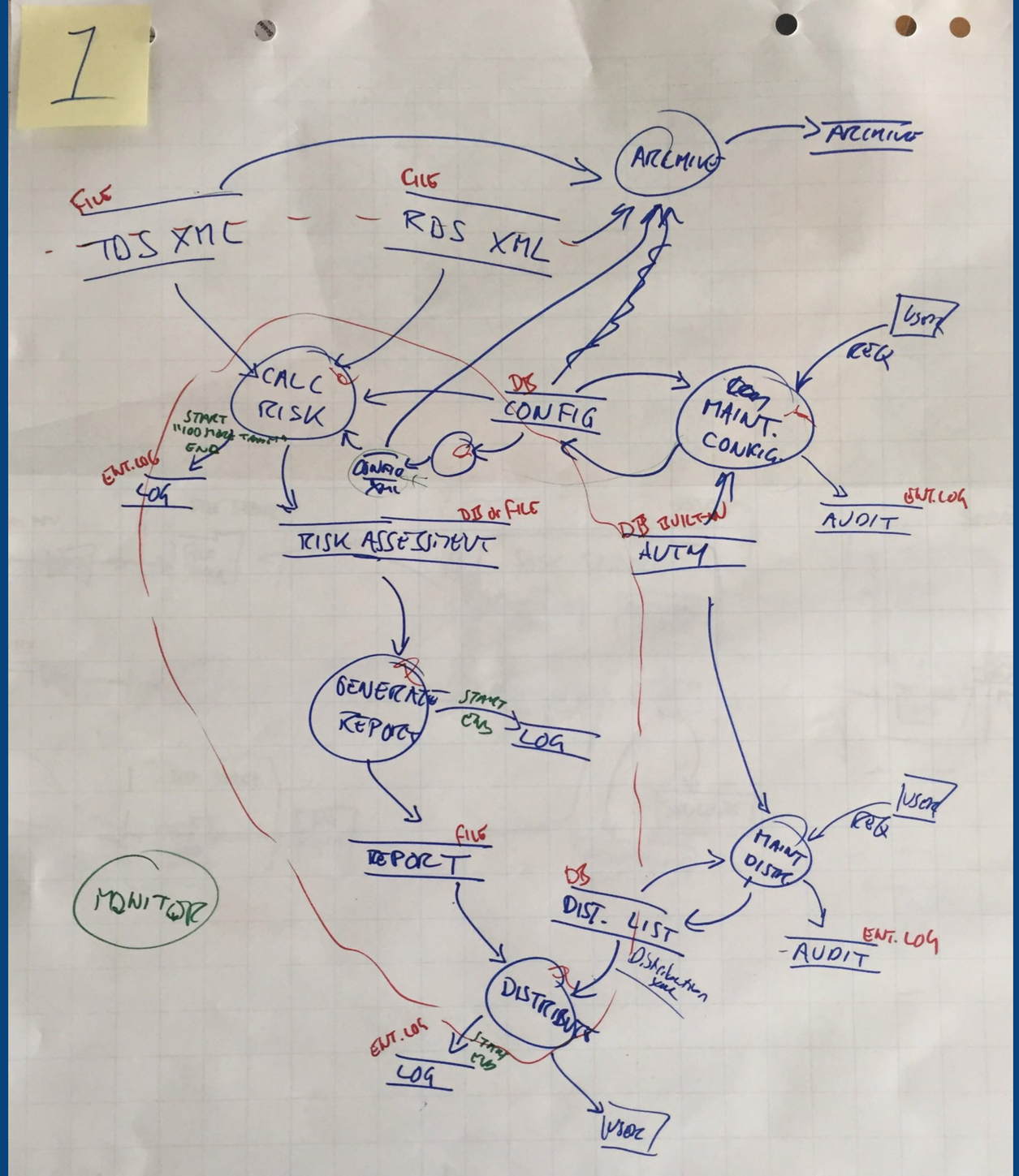

NO ANNOTATION ON FLOWS
SHOULD USE MORE COLORS
POST ITS CAN FALL OFF
What does colour mean?
—
MIXES DIFFERENT LEVELS OF DETAIL
NOT SURE OF TRANSITION BETWEEN DIFFERENT DIAGRAMS -
—
CONFLICTING LEVELS OF DETAIL IN PRESENTATION
What about the different arrows?
objects vs actions
WHY ARE SOME LINES PINK ?
WHAT DO THE SHAPE MEAN ?
UML IS GOOD, BUT NOT EVERYONE KNOWS IT
WHAT DO LINES RE-PRESENT? (DATA? CONTROL? DEP.?)
What shape mean
What's the DB-like icon?
Not sure what this is
DIFFERENT LEVELS IN SAME DIAGRAM
ARE THE ARROWS THE RIGHT WAY ROUND?

Moving fast in the same direction
as a team requires
good communication

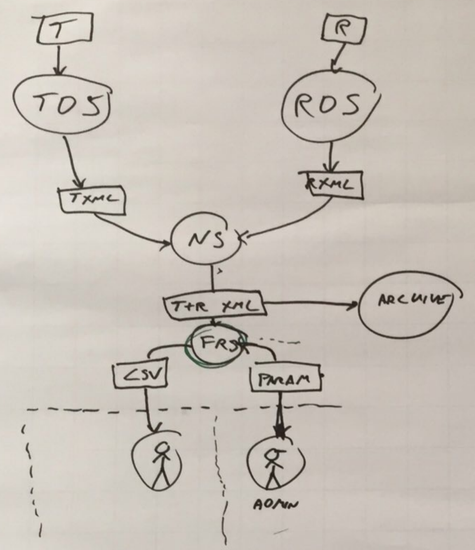
I’ve run this workshop
in **25+ countries**
for **10,000+ people**

Software architects
struggle to communicate
software architecture


~5pm NY
TDS
XML
FILE SERVER
FILE SHARE
poll
~5pm NY
RDS
XML
FILE SHARE
poll
FILE SERVER
FILE SHARE
Risk Report
Configure access rights
SERVER
RISK SERVICE
HEARTBEAT
CALCULATE RISK
GENERATE REPORT
SCHEDULER
LOG
SPLUNK
MONITORING
SERVER
RISK ASSESSMENT
RISK SVC CONFIG.
DISTRIB. LIST
Configure access rights
SERVER
CONFIG/ADMIN WEB APP
Config access rights
1

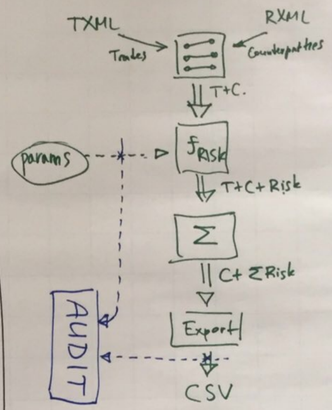

1
ARCHIVIO
FILE
TDS XML
FILE
RDS XML
CALC RISK
DB
CONFIG
MAINT. CONFIG
USER
REQ
START
END
ENT. LOG
LOG
DB or FILE
RISK ASSESSMENT
DB BUILT-IN
AUTH
AUDIT
ENT. LOG
GENERATE REPORT
START
END
LOG
FILE
REPORT
MONITOR
MAINT DISTR
USER
REQ
DB
DIST. LIST
AUDIT
ENT. LOG
DISTRIBUTE
ENT. LOG
START
END
LOG
USER

SA



AA

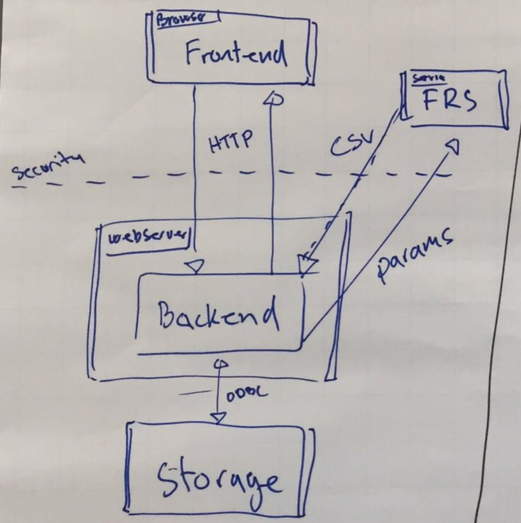
FRS



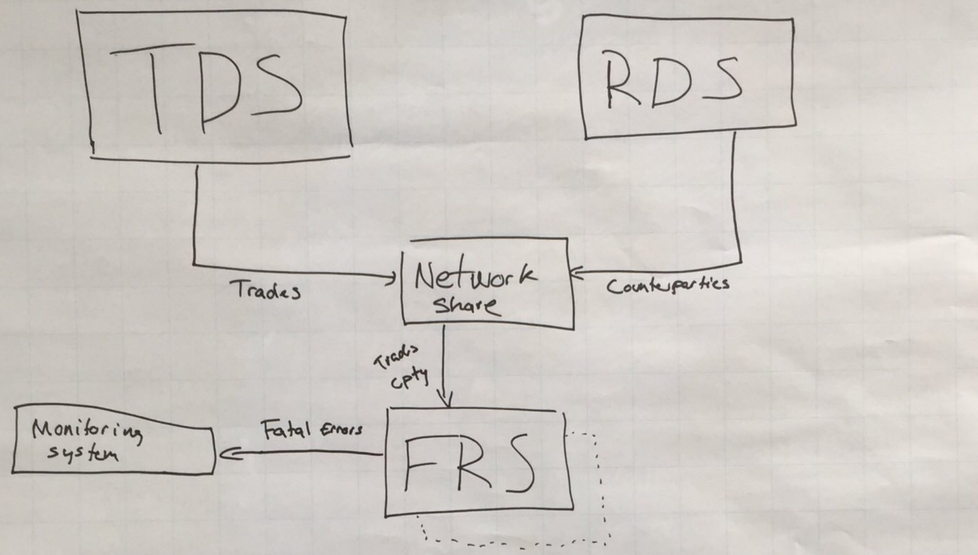
AA

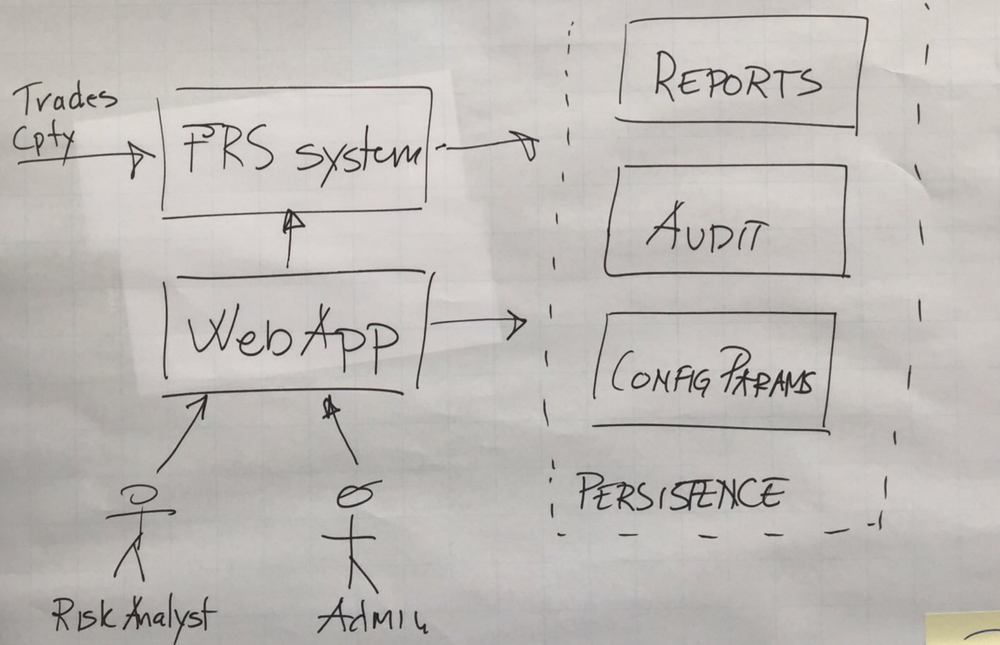
Components & Connectors view

Risk Viewer



2


TDS
RDS
Trades
Network share
Counterparties
Trade cpty
Monitoring system
Fatal errors
FRS


Trades cpty
FRS system
Web App
Risk Analyst
Admin
REPORTS
AUDIT
CONFIG PARAMS
PERSISTENCE

3

DB

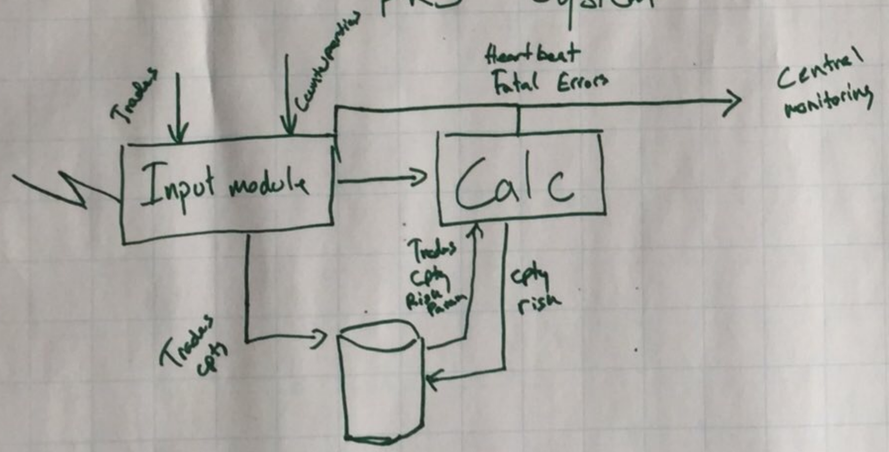
TRADES
TID (PK)
DATE
VALUE
CID (FK)

CPTYS
CID (PK)
NAME
ADDRESS

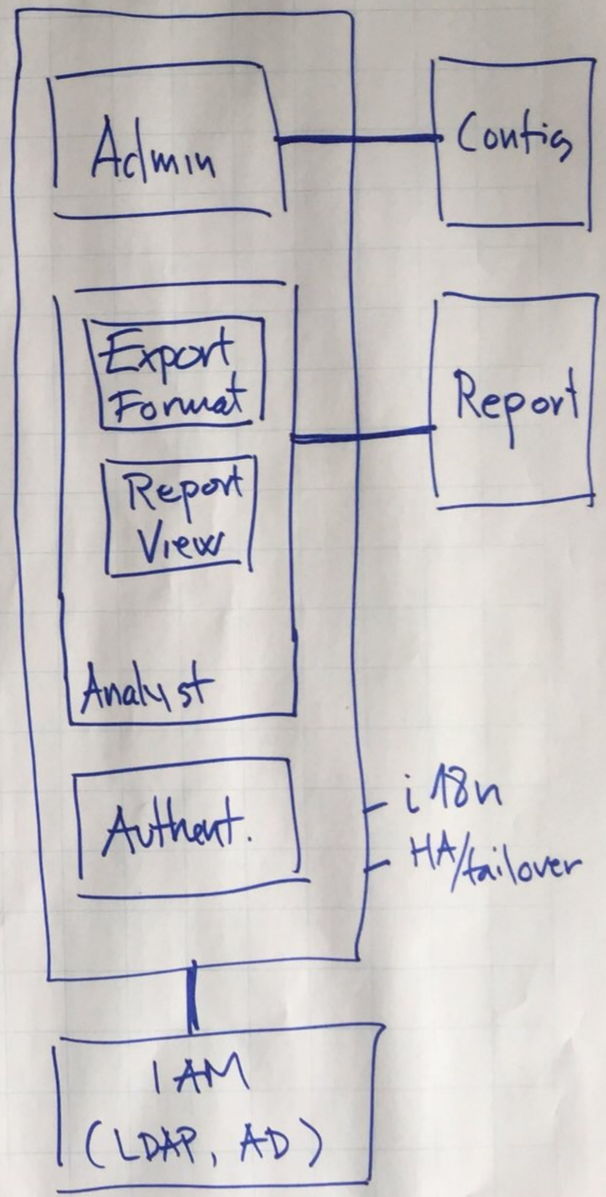
REPORTS
RID (PK)
CID (FK)
PID (FK)
DATE
RISK VALUE

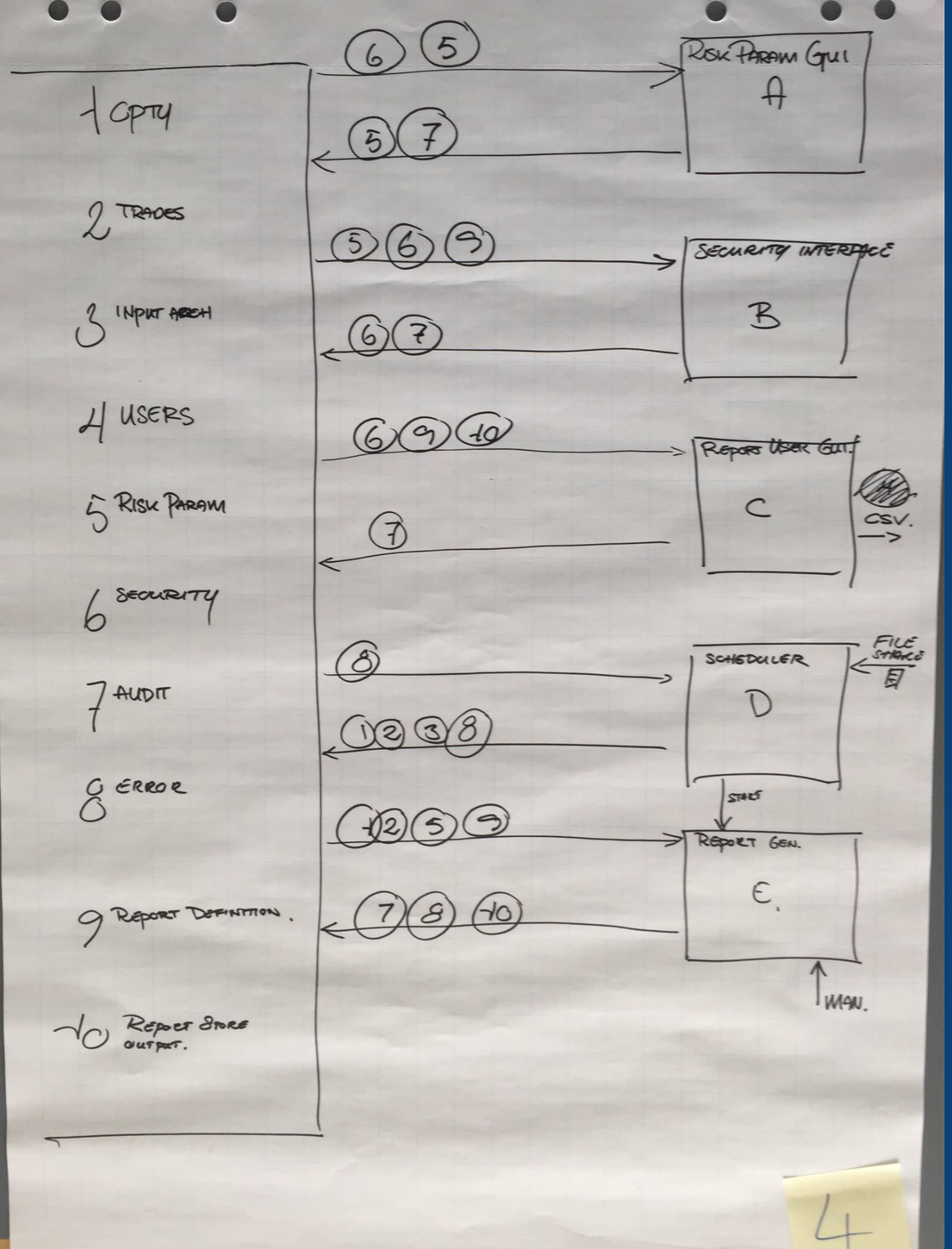
PARAMETERS
PID (PK)
RISK_PARAM_1
RISK_PARAM_2
DATE
LDAP_UID

FRS System



WebApp

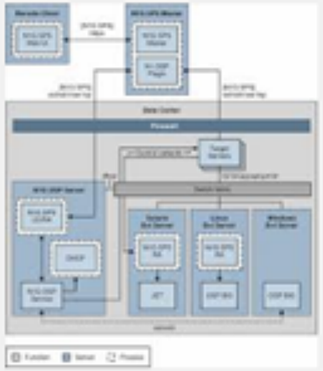
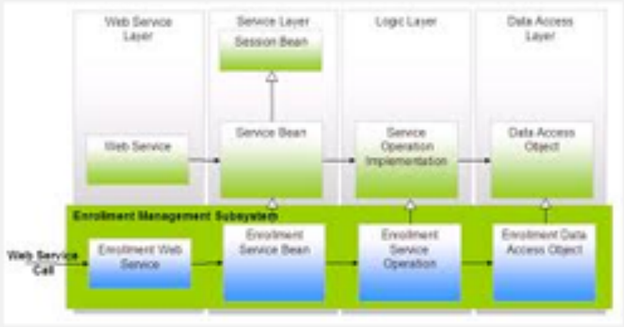


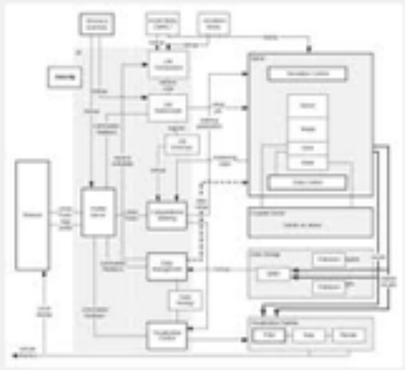
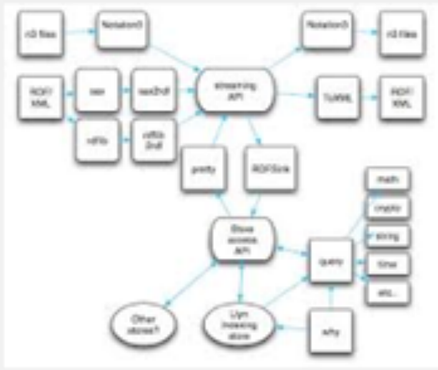

1 CPTY
2 TRADES
3 INPUT ARCH
4 USERS
5 RISK PARAM
6 SECURITY
7 AUDIT
8 ERROR
9 REPORT DEFINITION.
10 REPORT STORE OUTPUT.
RISK PARAM GUI
A
SECURITY INTERFACE
B
REPORT USER GUI
C
CSV.
SCHEDULER
D
FILE STORAGE
STATS
REPORT GEN.
E.
MAN.

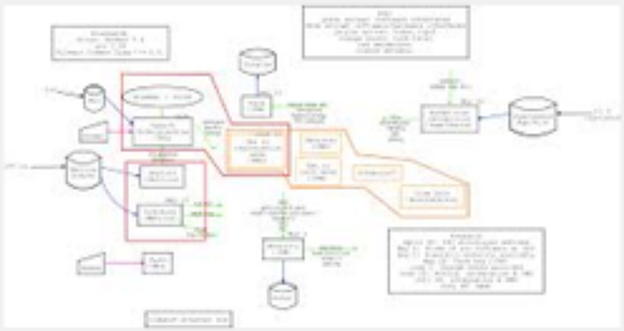
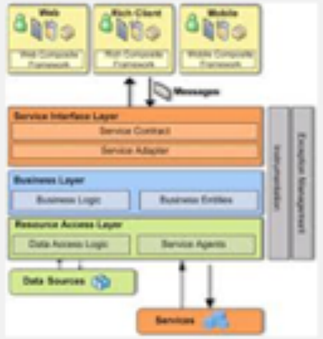
Google software architecture diagram

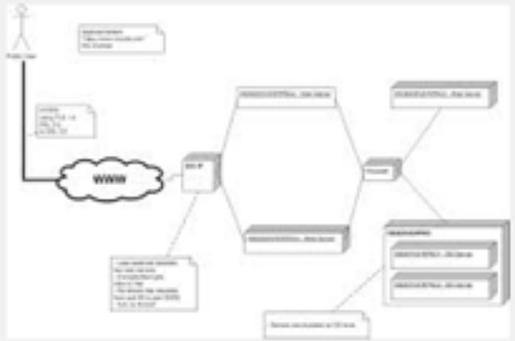
Sign in

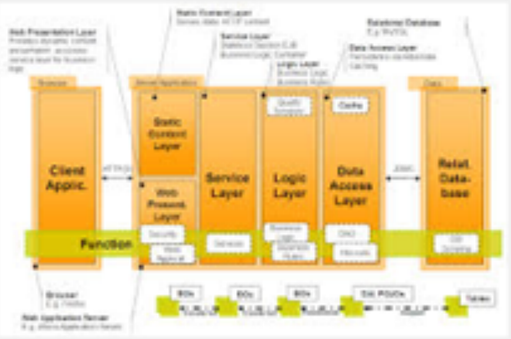
Web Images Videos Shopping News More Search tools SafeSearch

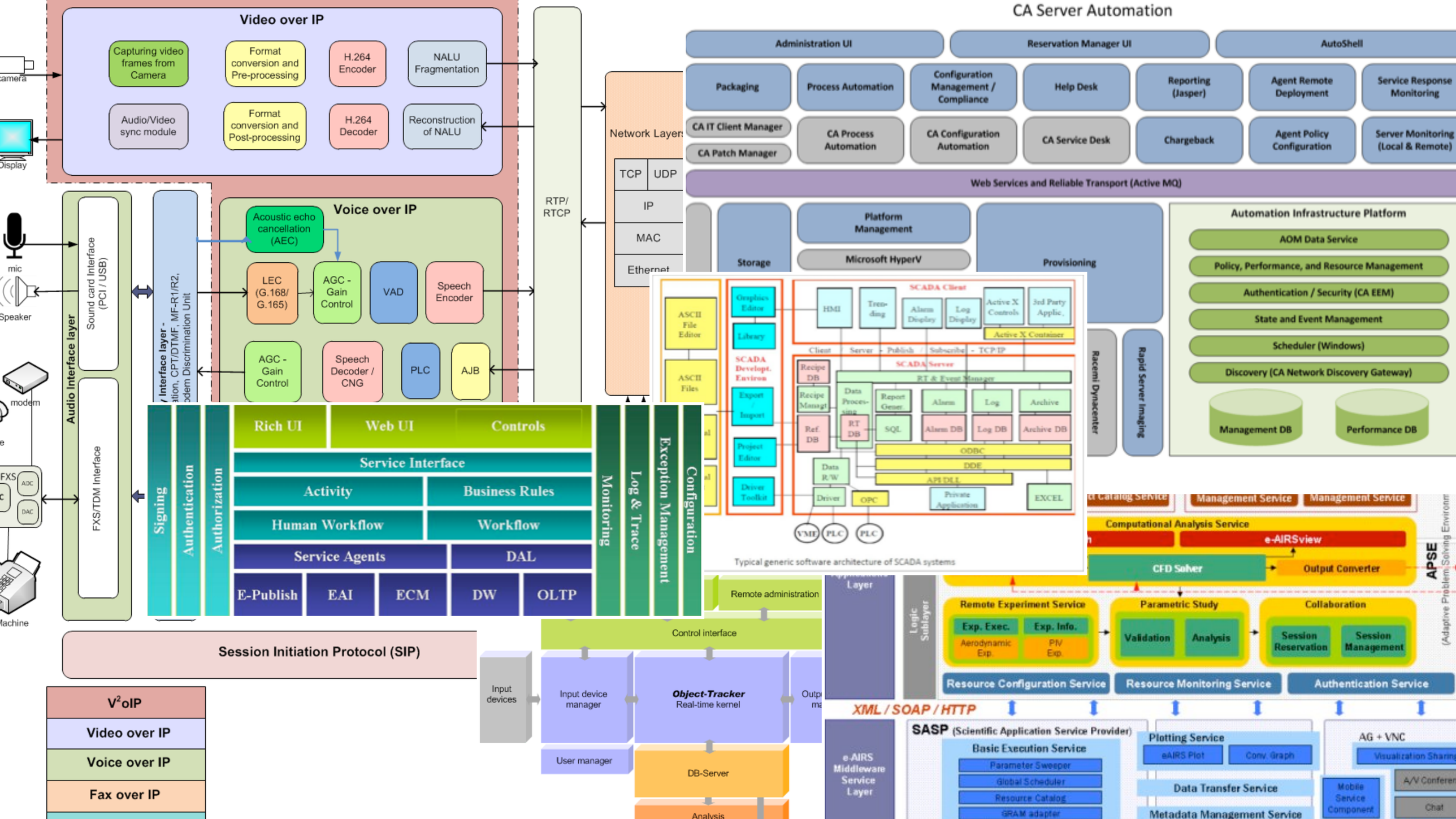



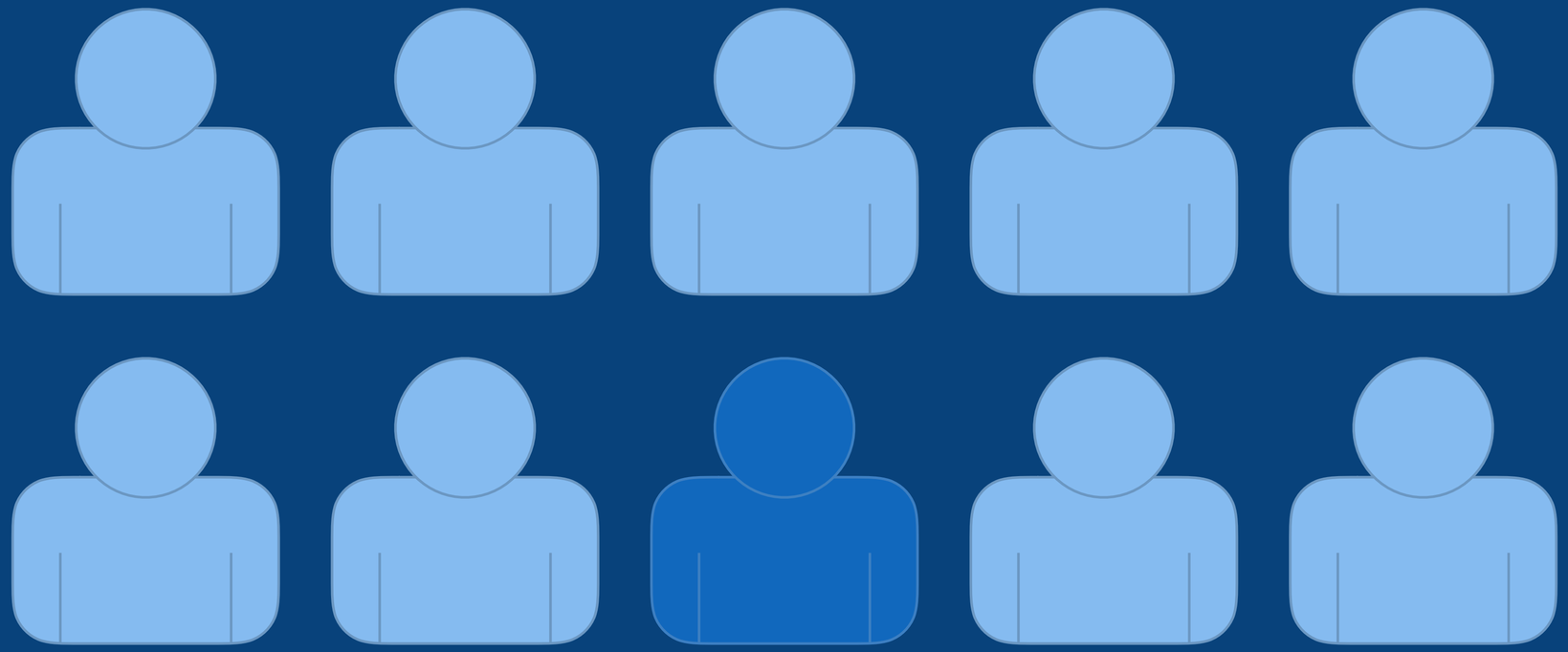




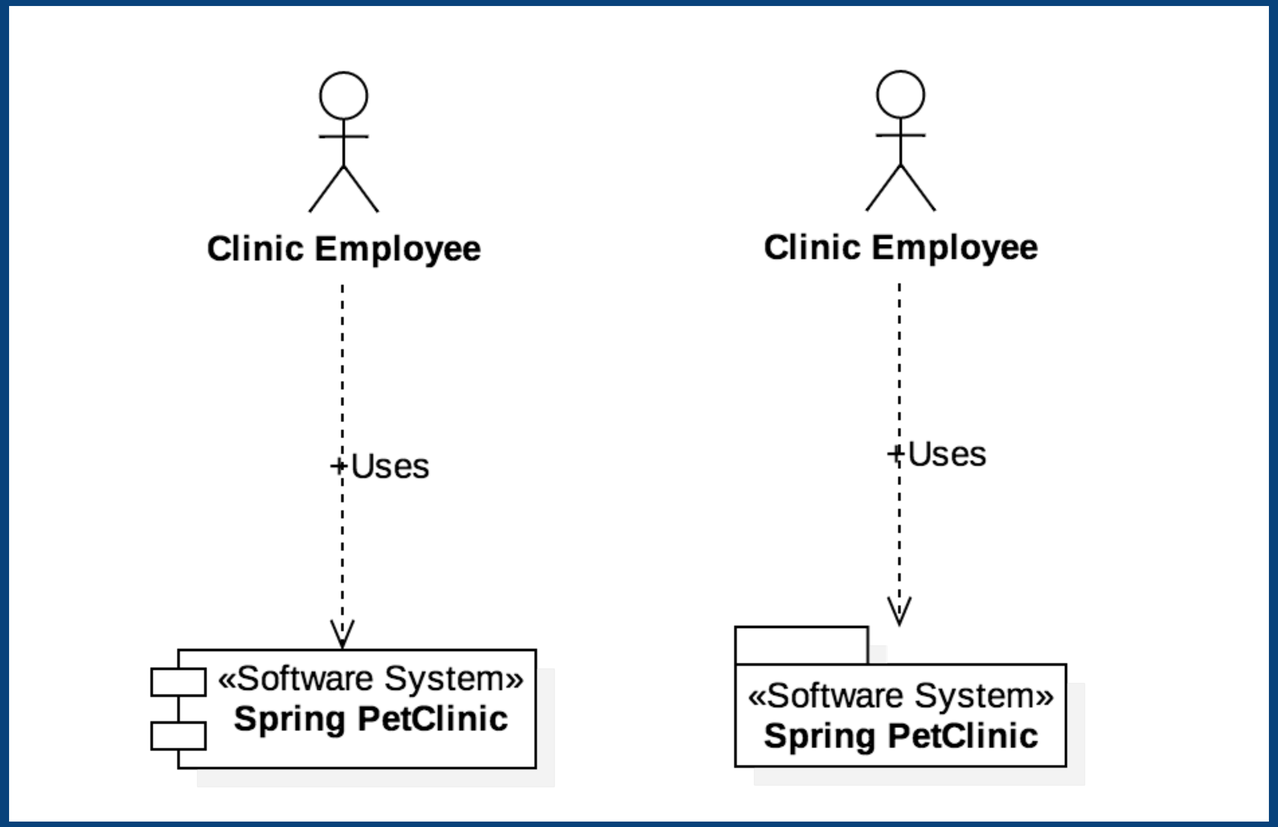


Video over IP
Capturing video frames from Camera
Format conversion and Pre-processing
H.264 Encoder
NALU Fragmentation
Audio/Video sync module
Format conversion and Post-processing
H.264 Decoder
Reconstruction of NALU
camera
Display
Voice over IP
Acoustic echo cancellation (AEC)
LEC (G.168/ G.165)
AGC - Gain Control
VAD
Speech Encoder
AGC - Gain Control
Speech Decoder / CNG
PLC
AJB
mic
Speaker
modem
FXS
ADC
DAC
Machine
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 Patch Manager
CA Process Automation
CA Configuration Automation
CA Service Desk
Chargeback
Agent Policy Configuration
Server Monitoring (Local & Remote)
Web Services and Reliable Transport (Active MQ)
Storage
Platform Management
Microsoft HyperV
Provisioning
Racemi Dynacenter
Rapid Server Imaging
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
ASCII File Editor
ASCII Files
Graphics Editor
Library
SCADA Develop. 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 Manager
Data Processing
Report Generator
RT & Event Manager
Alarm
Log
Archive
Ref. DB
RT DB
SQL
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
Layer
Logic Sublayer
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
XML / SOAP / HTTP
e-AIRS Middleware Service Layer
SASP (Scientific Application Service Provider)
Basic Execution Service
Parameter Sweeper
Global Scheduler
Resource Catalog
GRAM adapter
Plotting Service
eAIRS Plot
Conv. Graph
Data Transfer Service
Metadata Management Service
AG + VNC
Visualization Sharing
Mobile Service Component
A/V Conferen
Chat

Do **you** use UML?



In my experience, optimistically,

1 out of 10 people use UML



I do use UML, but not for software architecture

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.

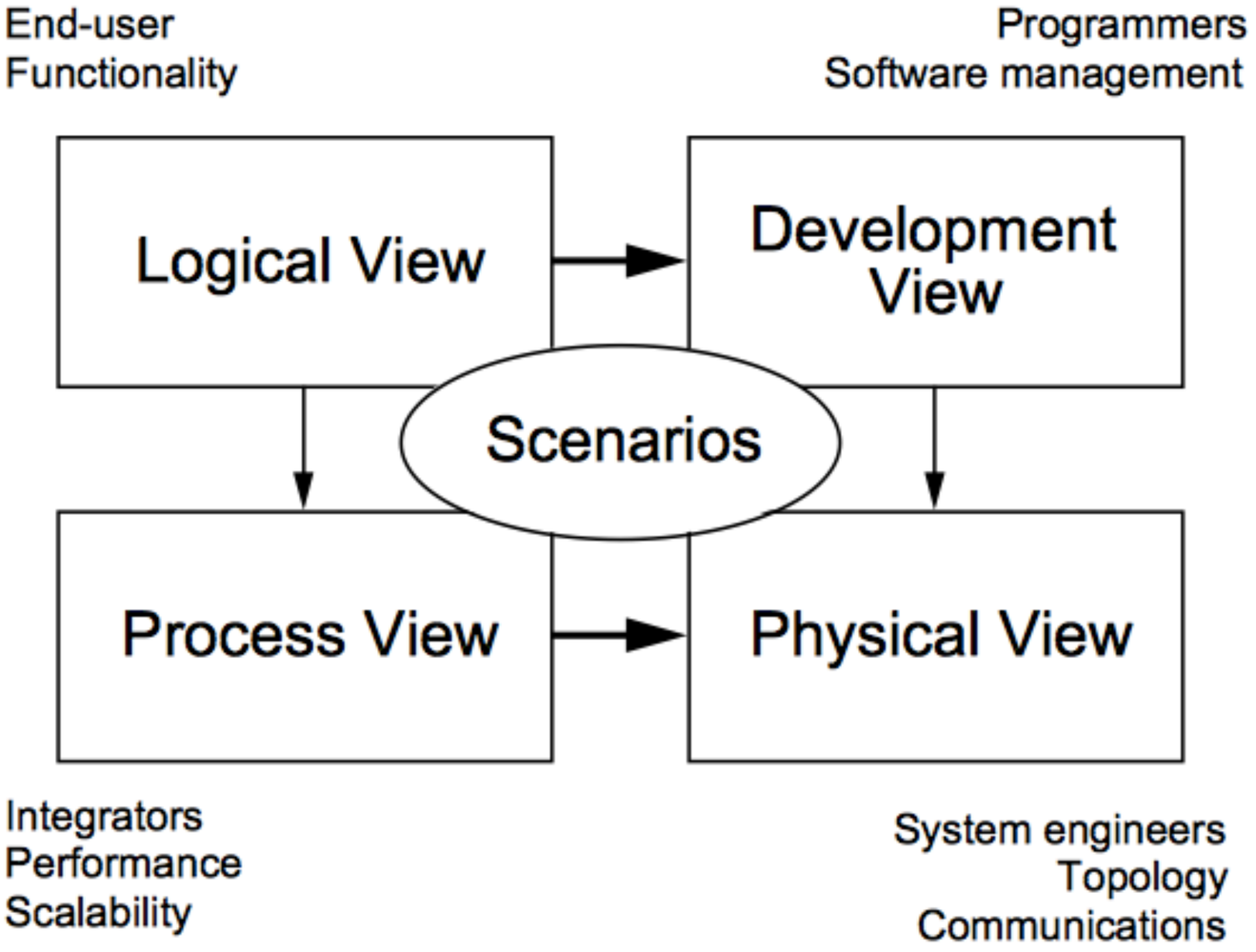
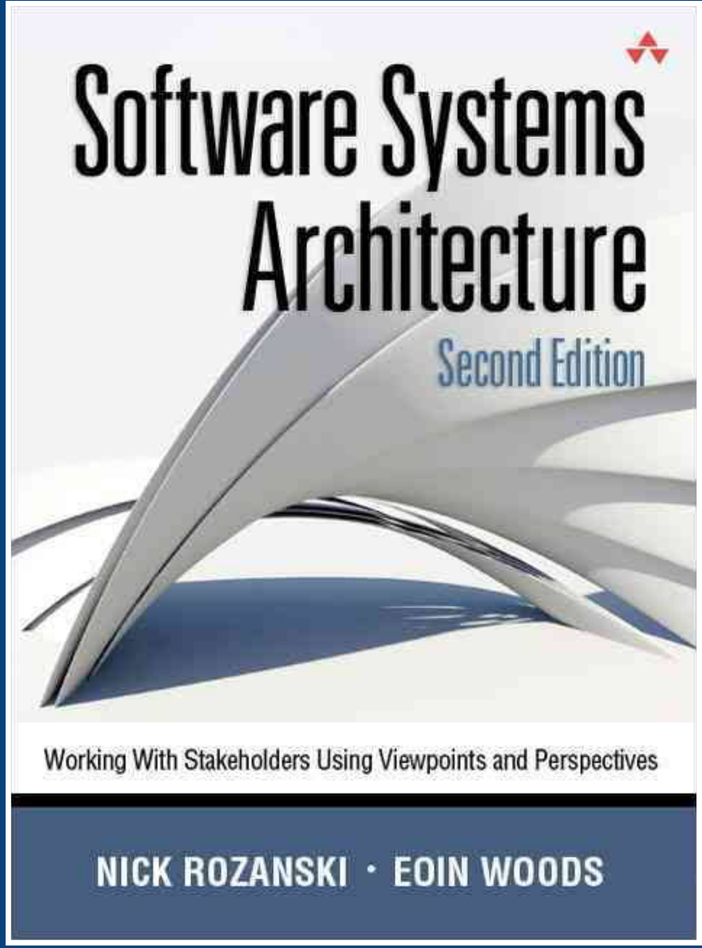
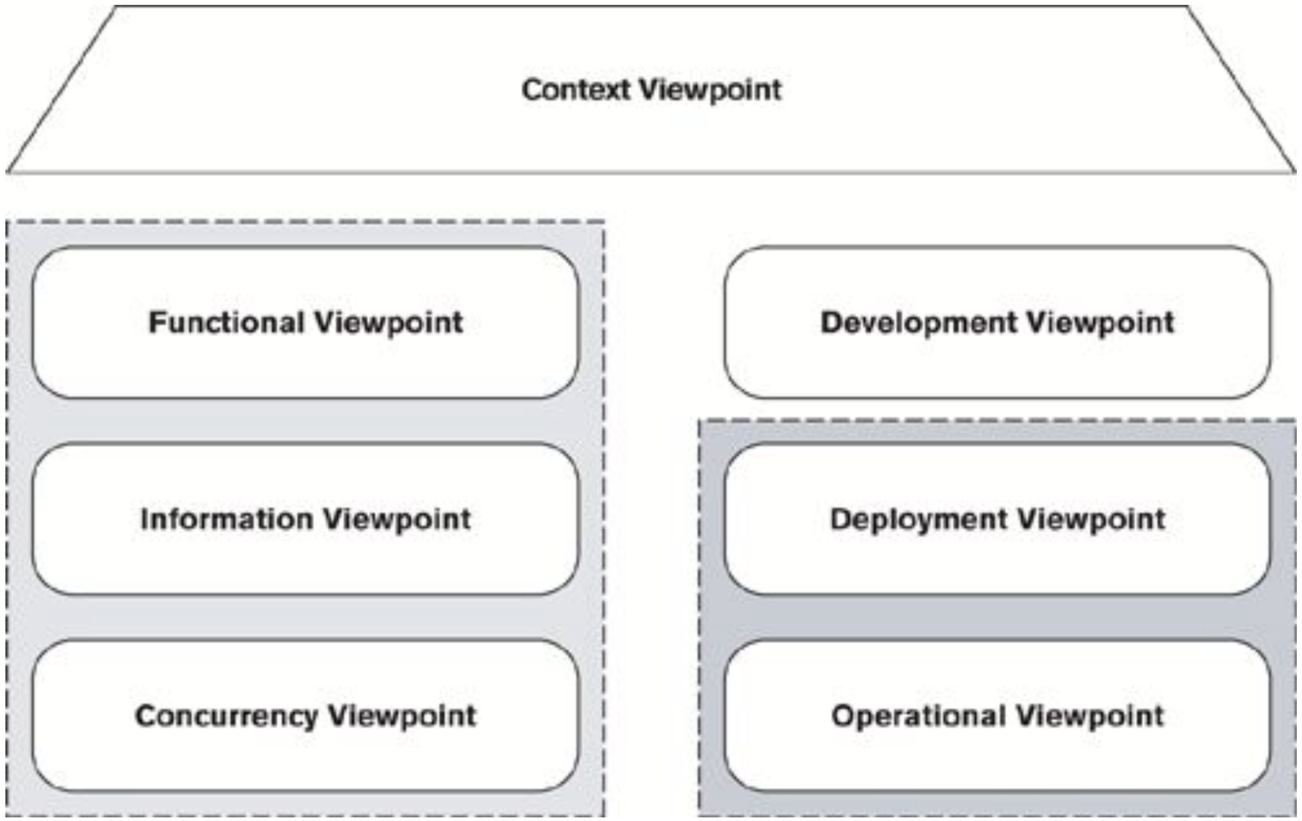


Figure 1 — The "4+1" view model

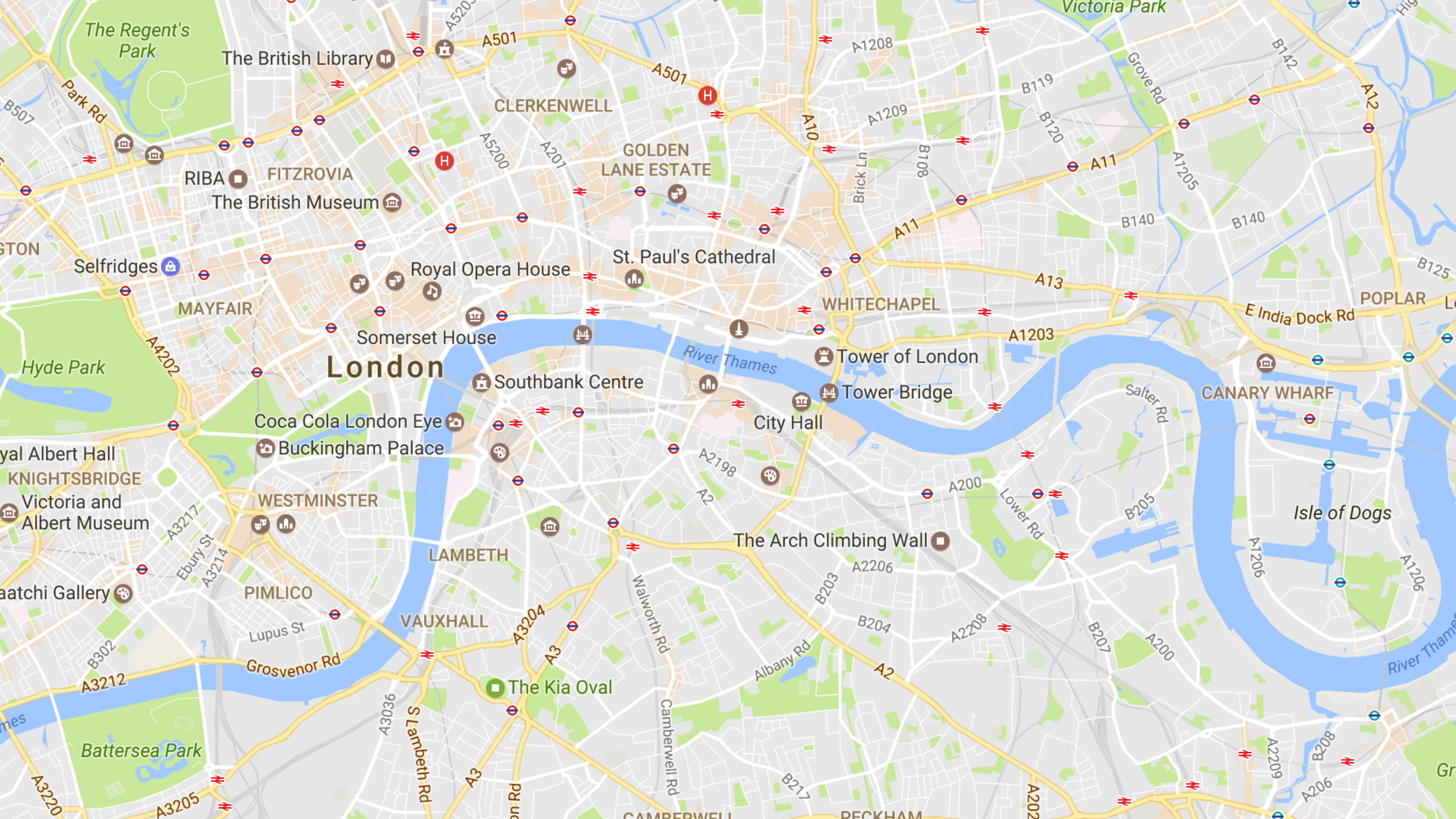




“Viewpoints and Perspectives”

Do the architecture diagrams accurately reflect what you are going to build?

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


The Regent's Park
The British Library
A501
CLERKENWELL
GOLDEN LANE ESTATE
RIBA
FITZROVIA
The British Museum
Park Rd
A5200
A201
A10
A1208
A1209
B119
B120
B108
Brick Ln
Grove Rd
Victoria Park
B142
A12
A11
A1205
B140
Selfridges
Royal Opera House
St. Paul's Cathedral
MAYFAIR
WHITECHAPEL
A13
POPLAR
E India Dock Rd
B125
A1203
Somerset House
London
River Thames
Tower of London
Tower Bridge
Southbank Centre
City Hall
CANARY WHARF
Salter Rd
Hyde Park
A4202
Coca Cola London Eye
Buckingham Palace
KNIGHTSBRIDGE
Victoria and Albert Museum
WESTMINSTER
A2198
A2
A200
Lower Rd
B205
Isle of Dogs
A3217
Ebury St
A3214
LAMBETH
The Arch Climbing Wall
A2206
B203
A1206
PIMLICO
VAUXHALL
Lupus St
A3204
A3
Walworth Rd
B204
A2208
B207
Grosvenor Rd
A3212
B302
The Kia Oval
Albany Rd
Battersea Park
A3036
S Lambeth Rd
Camberwell Rd
B217
A2209
B208
A3220
A3205
PECKHAM

TOWEL RING
ARMOIRE
TUB FILLER
NICHE

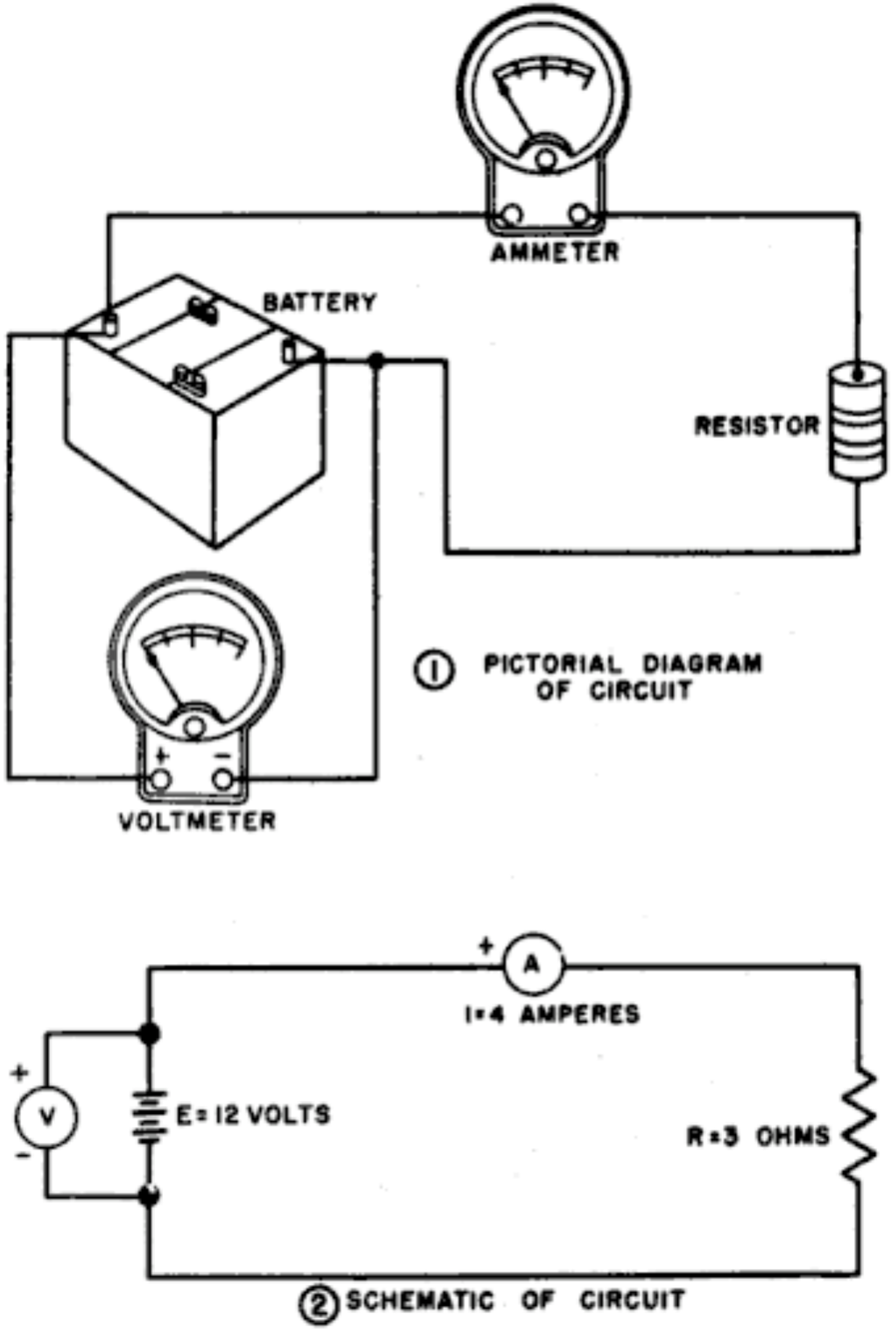
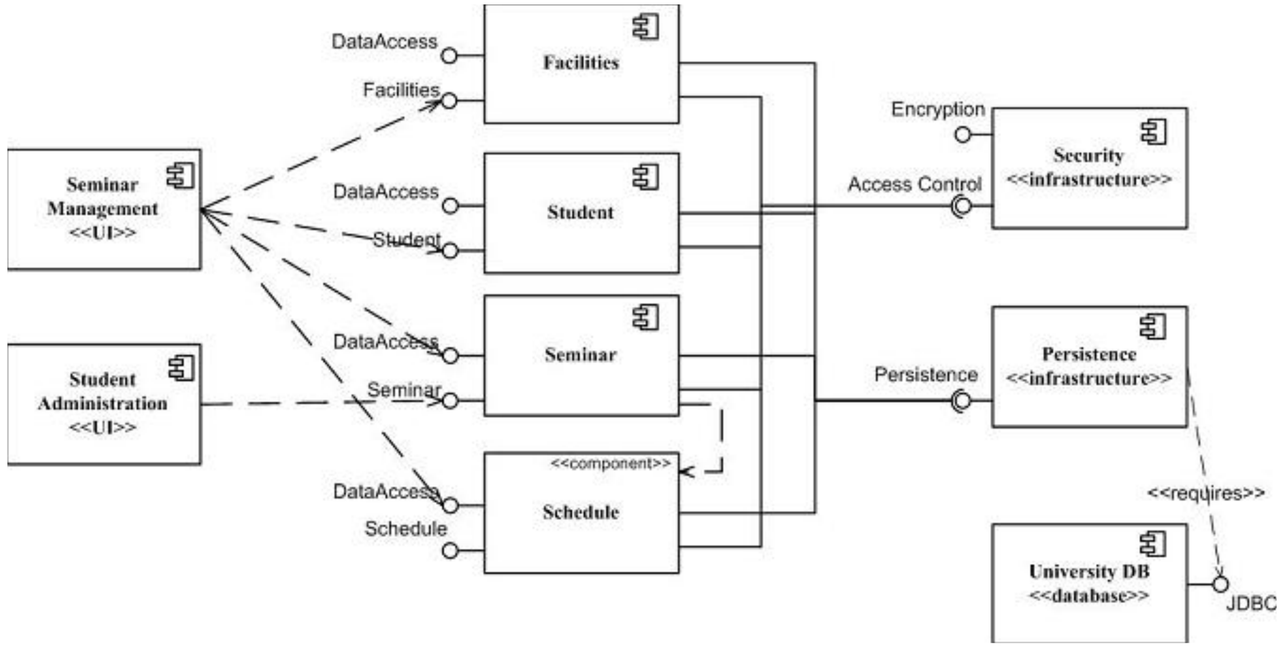
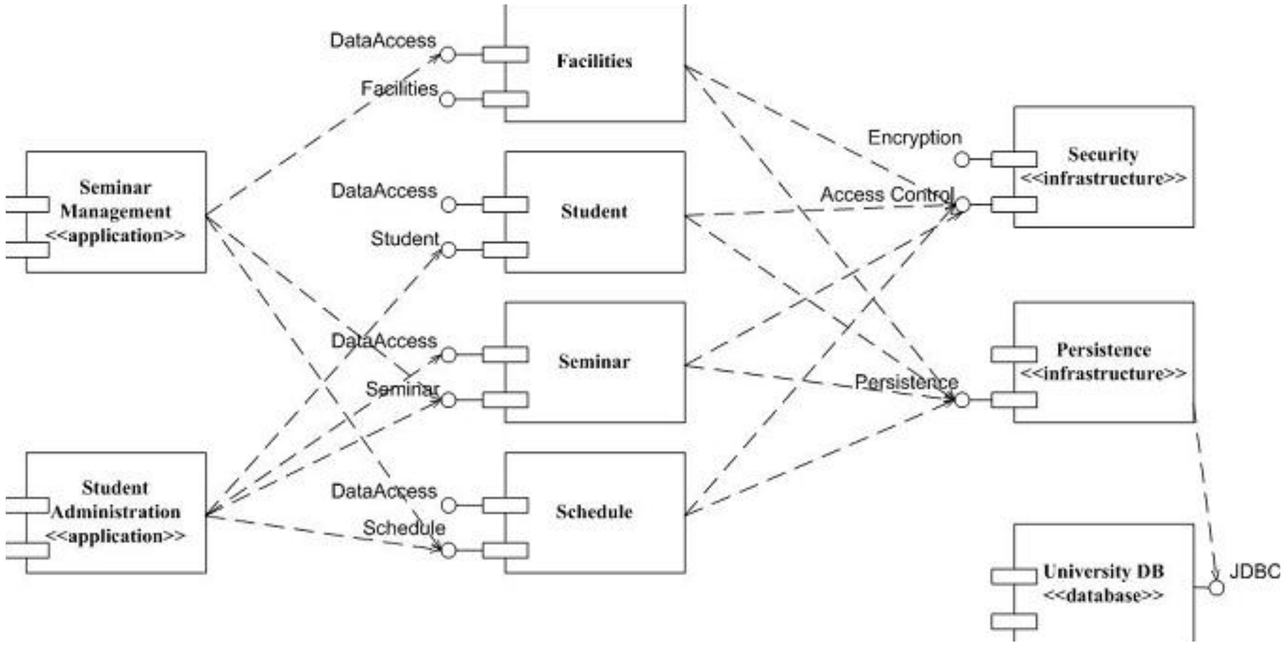
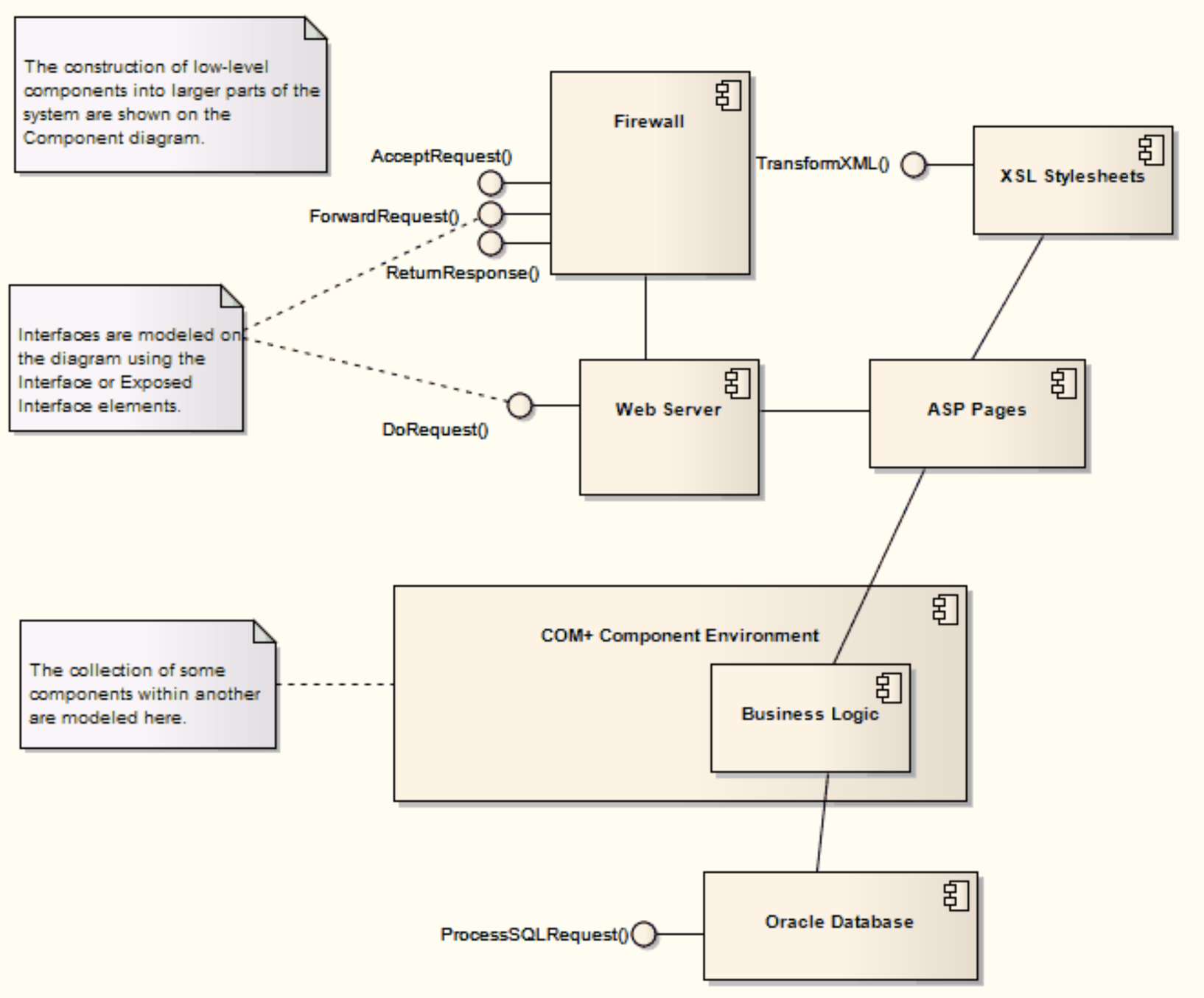
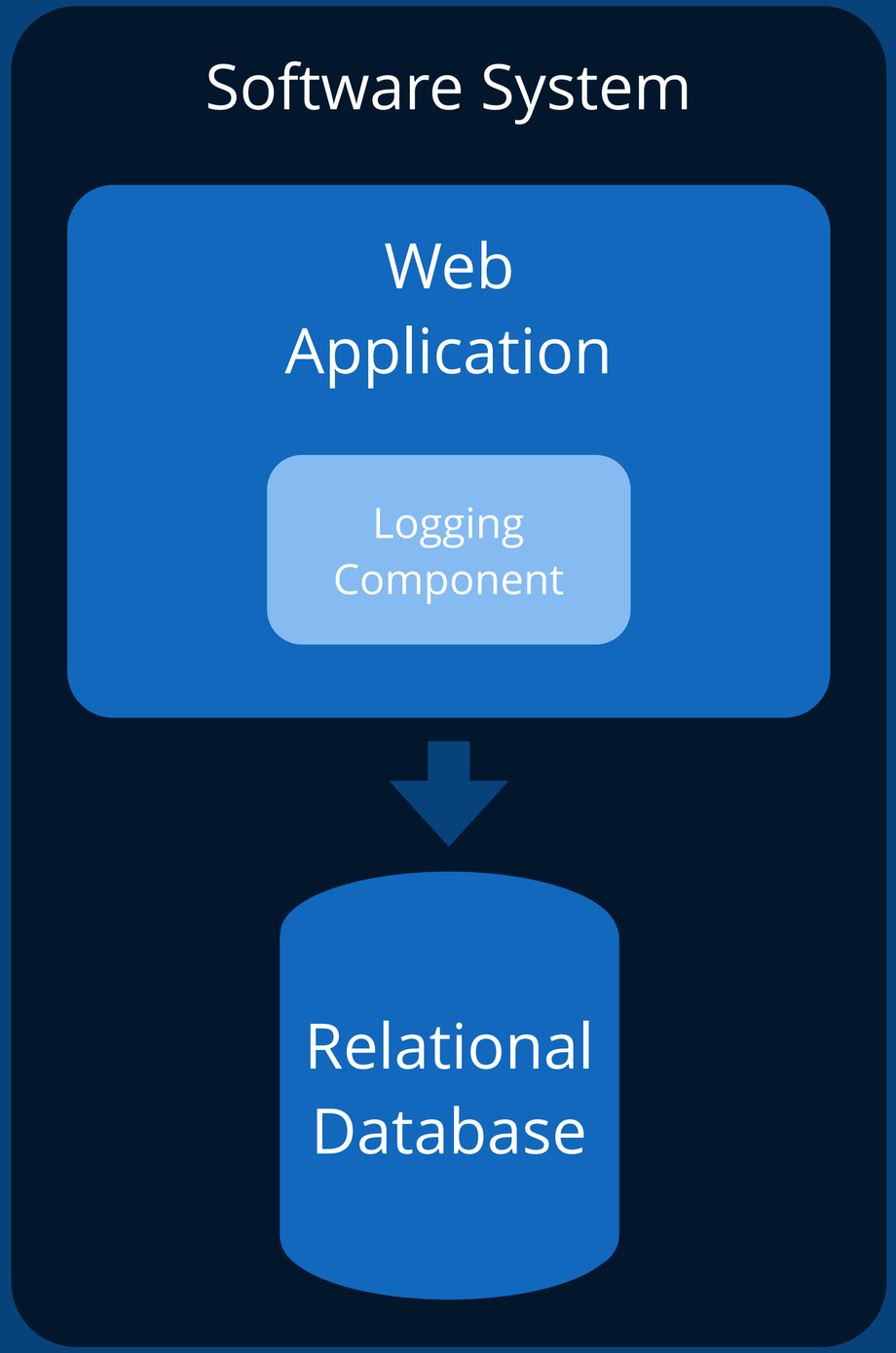


Figure 48. Diagram of a basic circuit.


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


DataAccess
Facilities
Facilities
Seminar Management <<application>>
DataAccess
Student
Student
Student Administration <<application>>
DataAccess
Seminar
Seminar
DataAccess
Schedule
Schedule
Encryption
Access Control
Security <<infrastructure>>
Persistence
Persistence <<infrastructure>>
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
COM+ Component Environment
The collection of some components within another are modeled here.
Business Logic
ProcessSQLRequest()
Oracle Database



[1] component

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

Simple Definition of COMPONENT

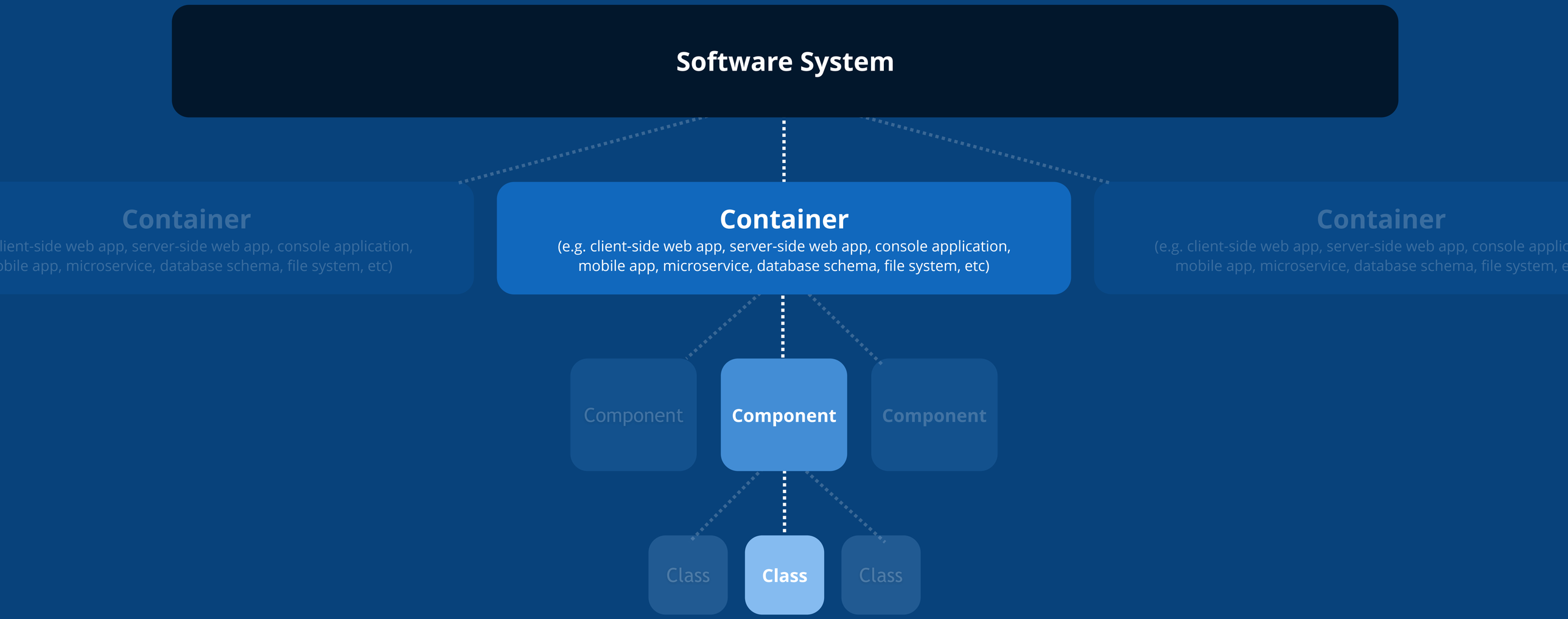
Popularity: Top 30% of words

: 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

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 **classes** (or **code**).

C4

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

Titles

Short and meaningful, numbered if diagram order is important

Orientation

Most important thing in the middle; be consistent across diagrams

Layout

Sticky notes and index cards make a great substitute for drawn boxes, especially early on

Labels

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

Lines

Favour unidirectional arrows, add descriptive text to provide additional information

Colour

Ensure that colour coding is made explicit; watch out for colour-blindness and black/white printers

Shapes

Don’t assume that people will understand what different shapes are being used for

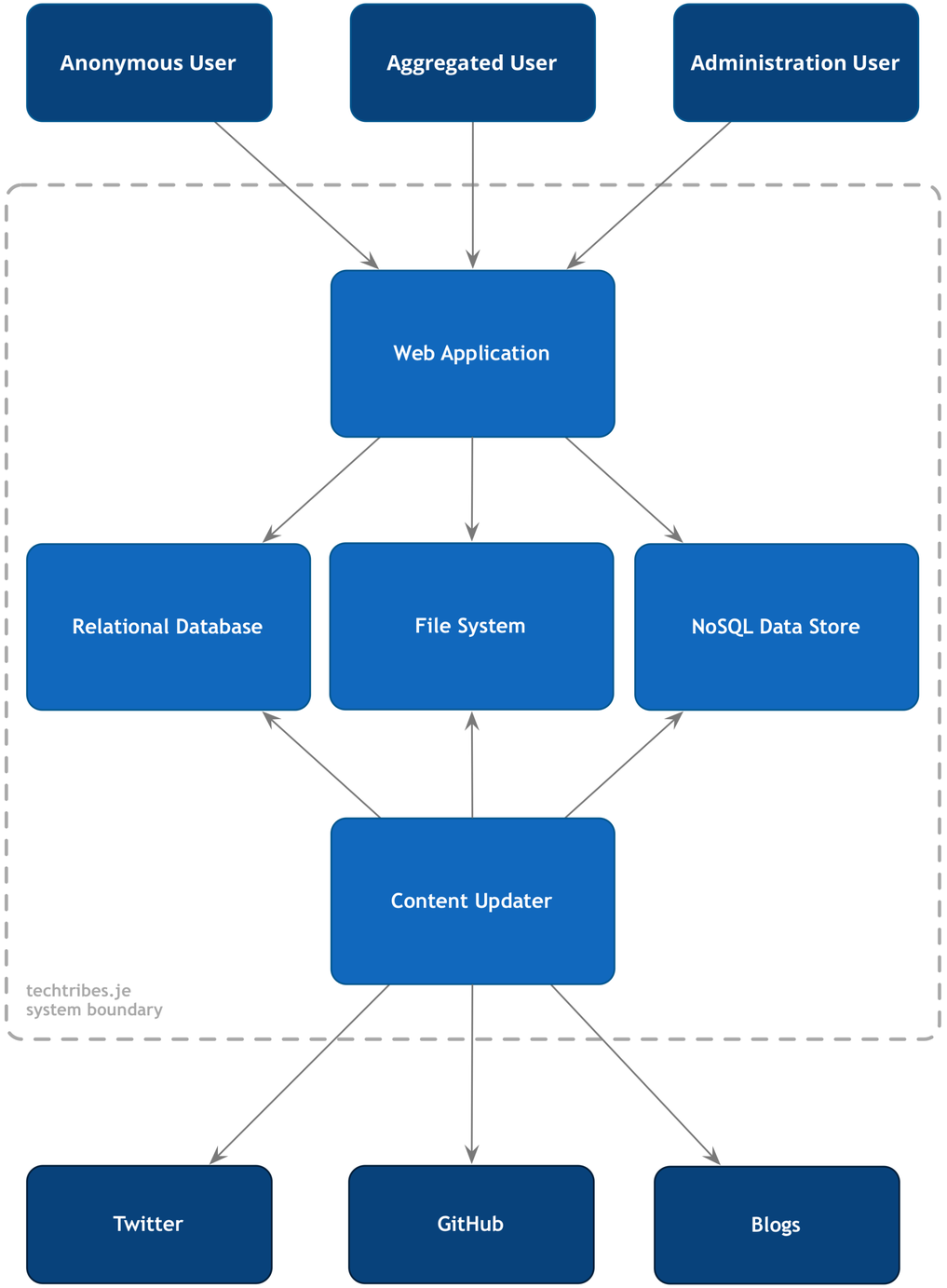
Keys

Explain shapes, lines, colours, borders, acronyms, etc

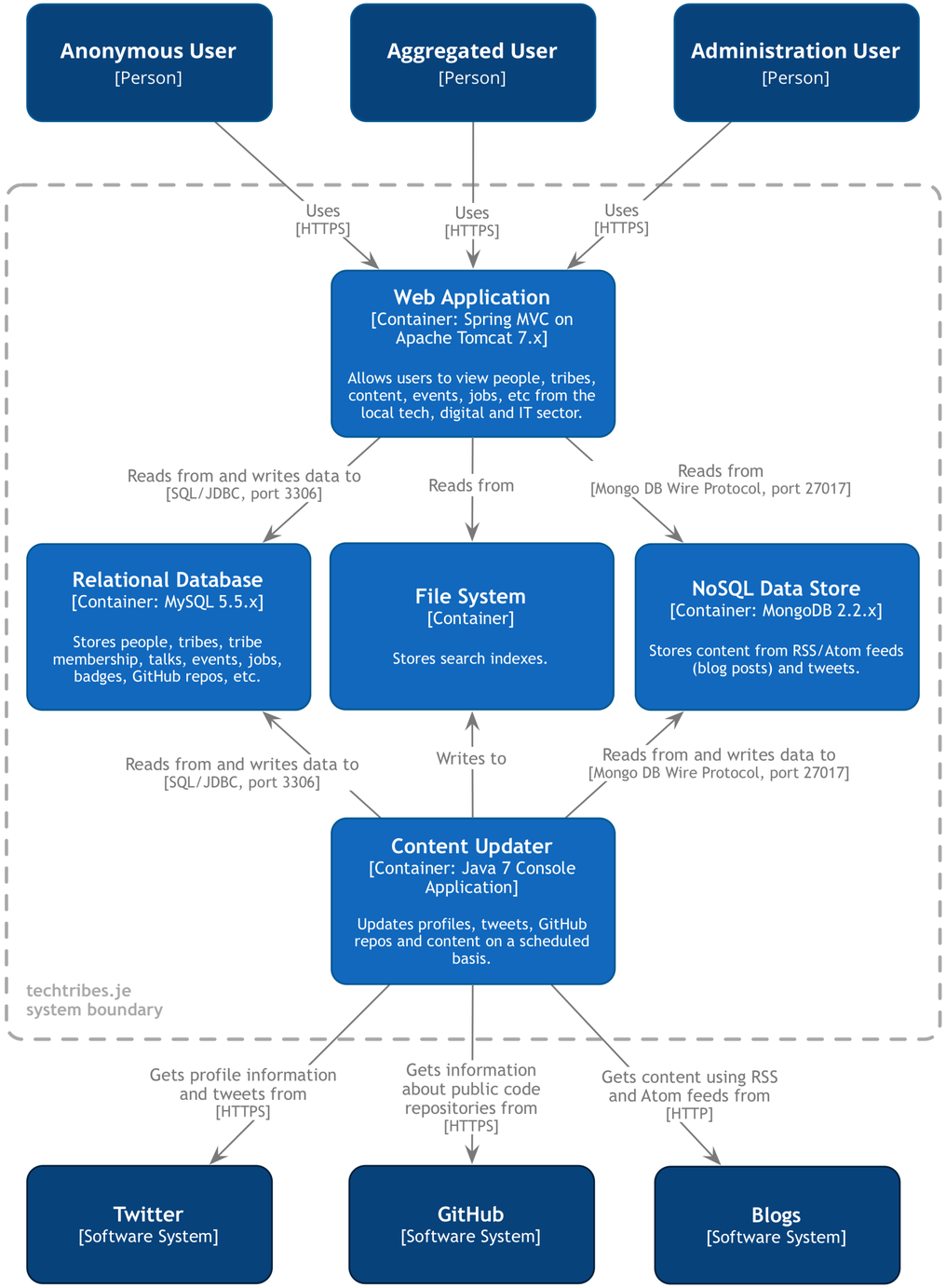
Responsibilities

Adding responsibilities to boxes can provide a nice “at a glance” view (Miller’s Law; 7±2)

Think about notation



[Containers] techtribes.je



[Containers] techtribes.je

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

techtribes.je

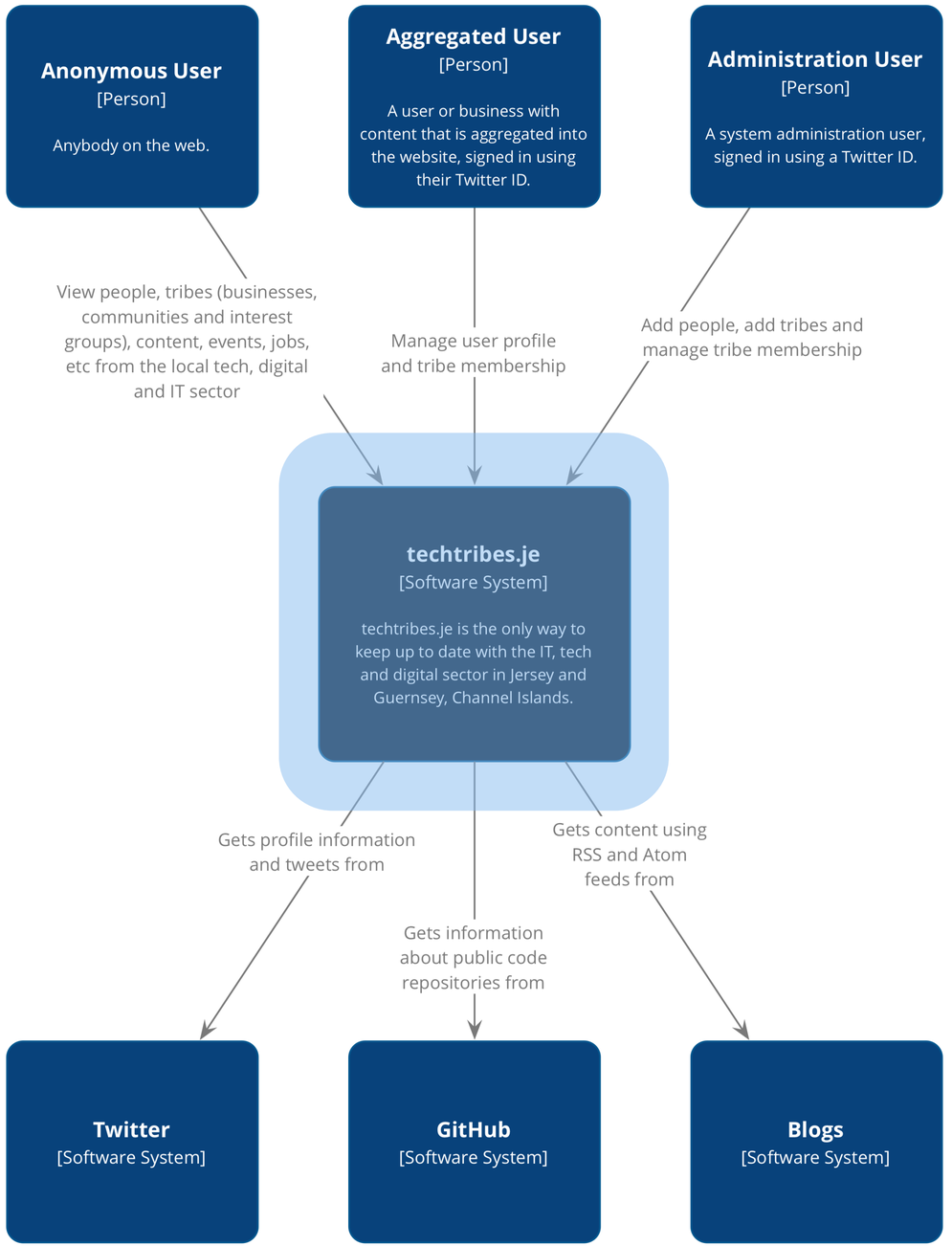
A simple content aggregator for the local tech and digital industry

1. System Context diagram

2. Container diagram

3. Component diagram

4. Class diagram



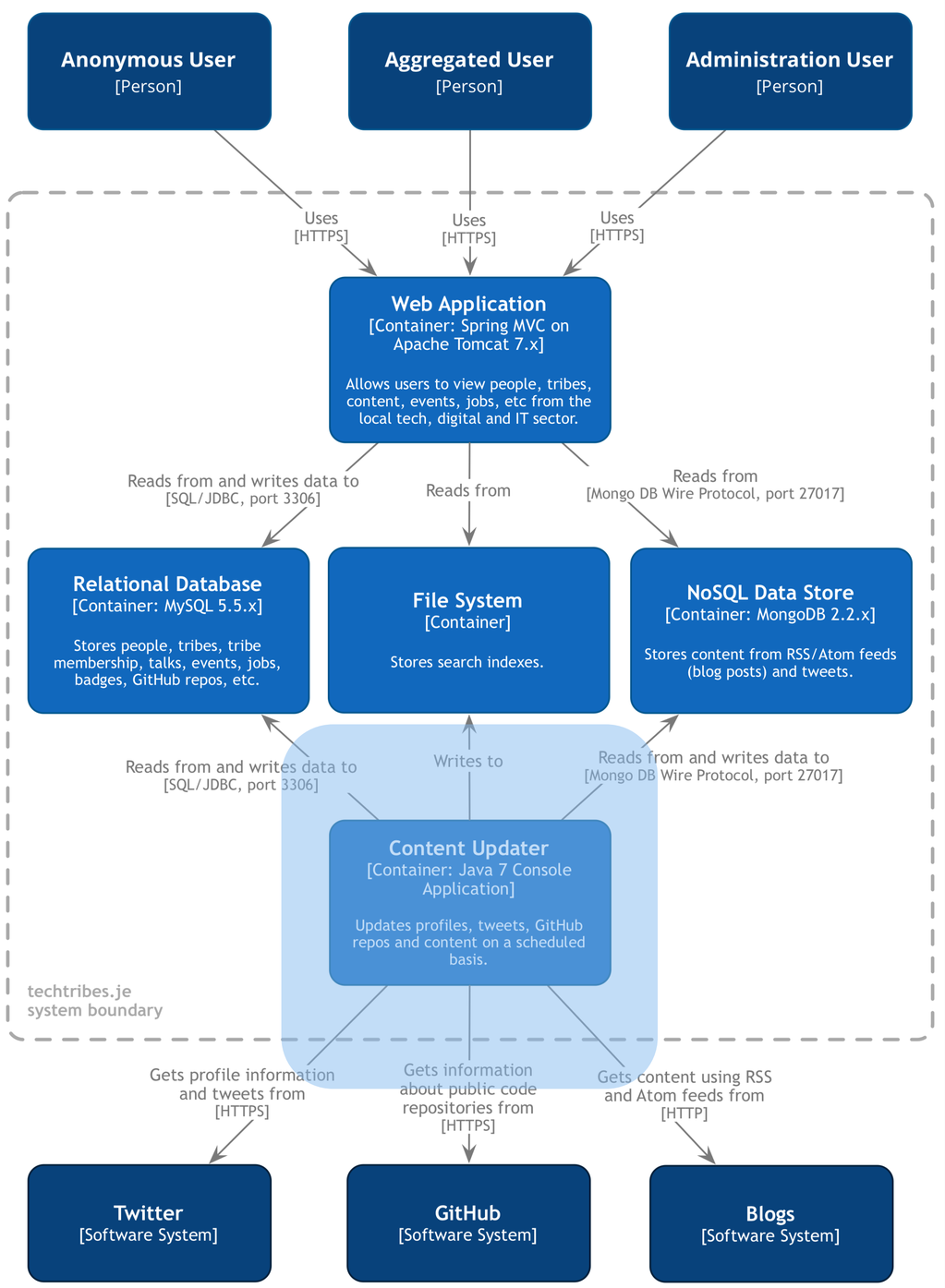
[System Context] techtribes.je

1. System Context diagram

2. Container diagram

3. Component diagram

4. Class diagram



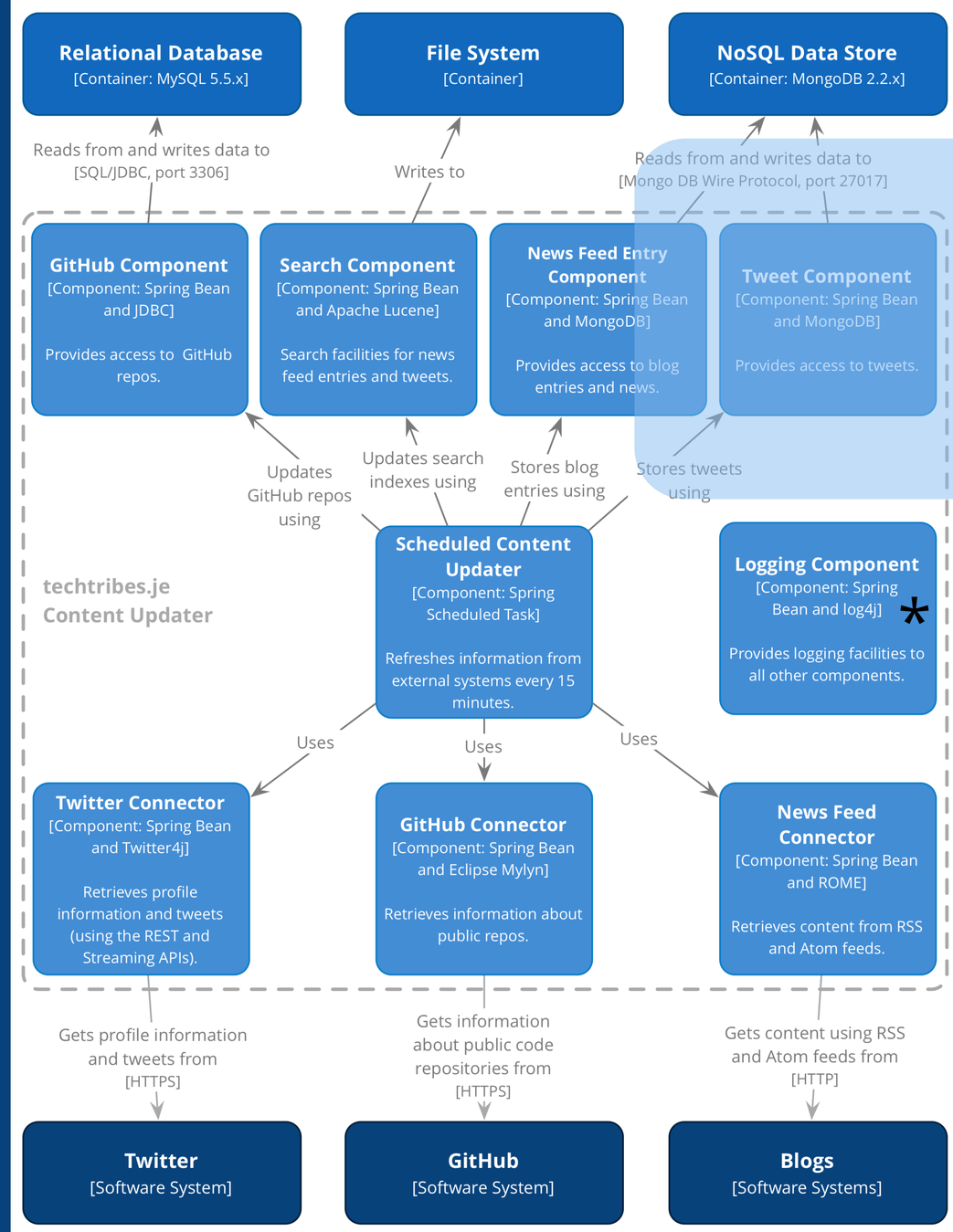
[Containers] techtribes.je

1. System Context diagram

2. Container diagram

3. Component diagram

4. Class diagram



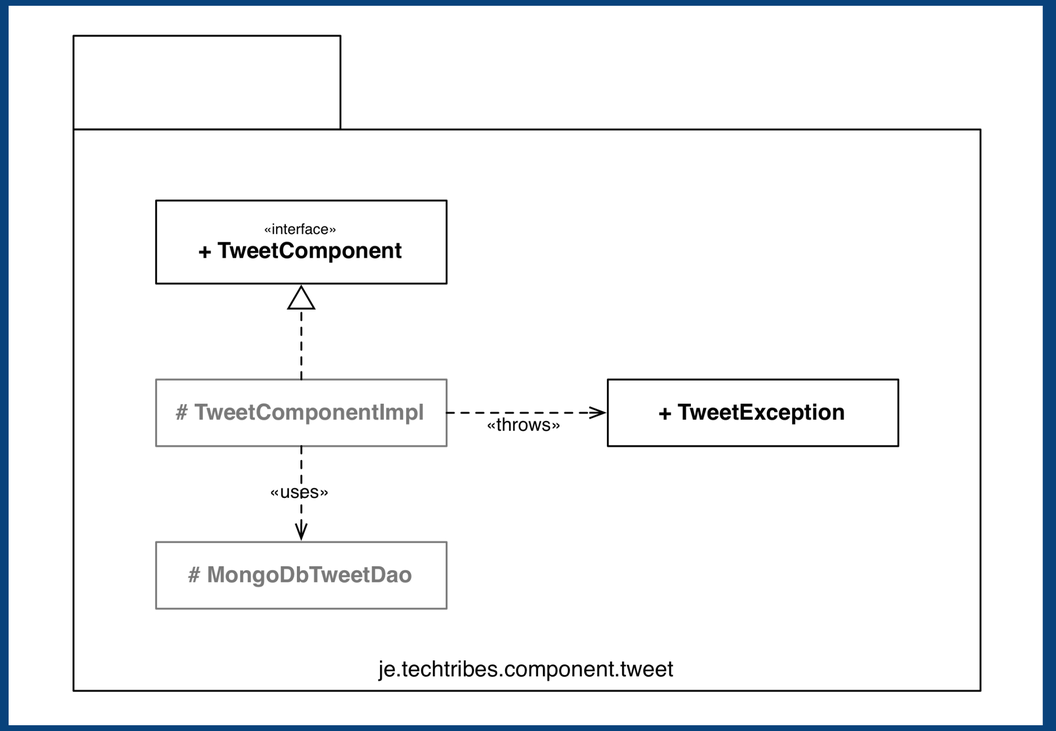
[Components] techtribes.je - Content Updater

1. System Context diagram

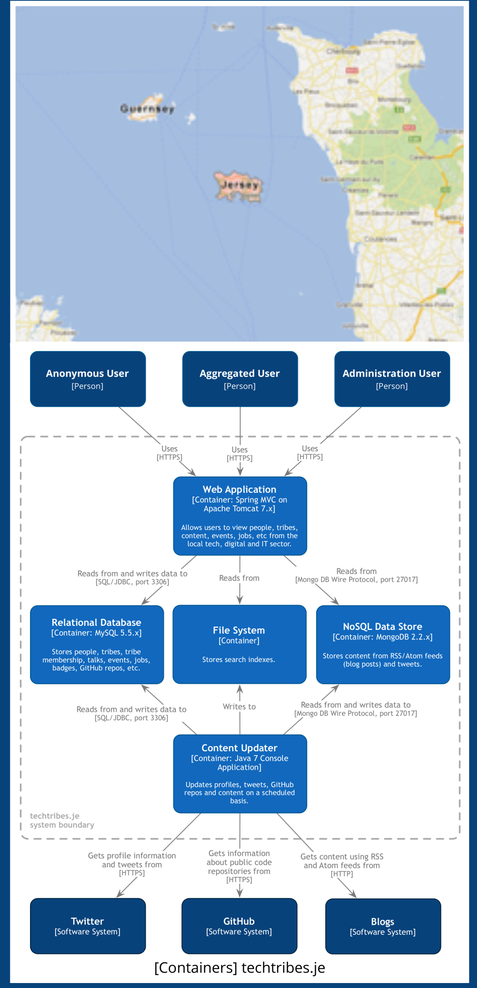
2. Container diagram

3. Component diagram

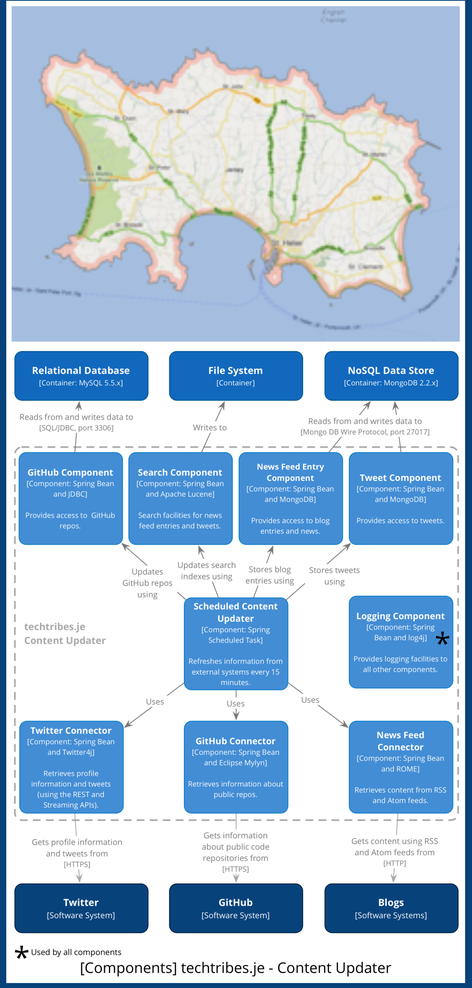
4. Class diagram



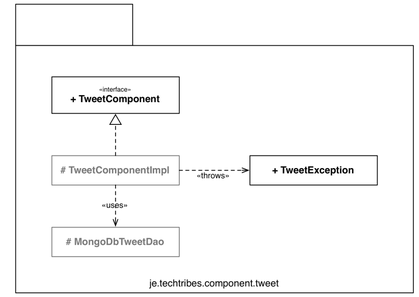
[System Context] techtribes.je



[Containers] techtribes.je

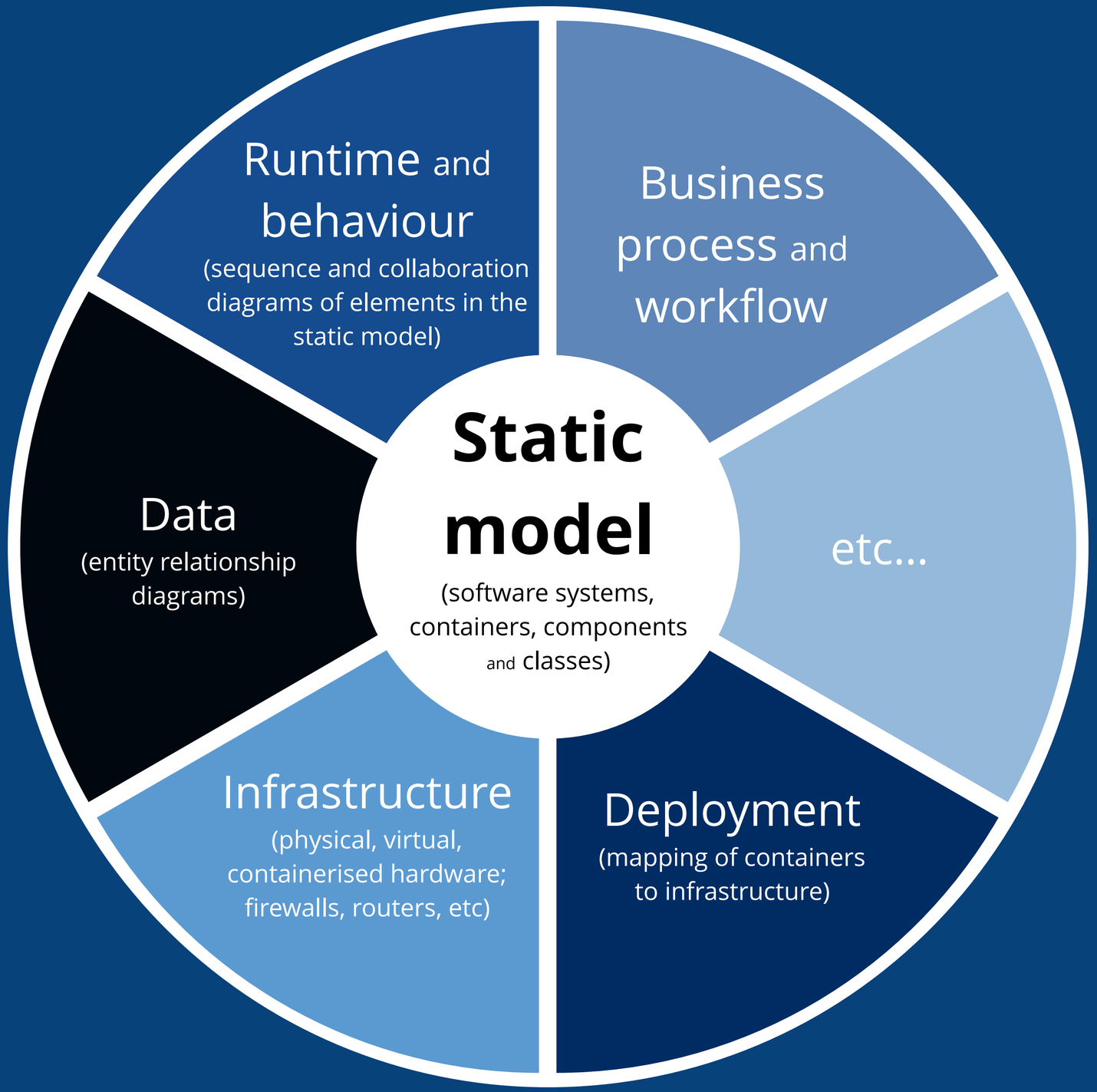


[Components] techtribes.je - Content Updater



Diagrams are maps

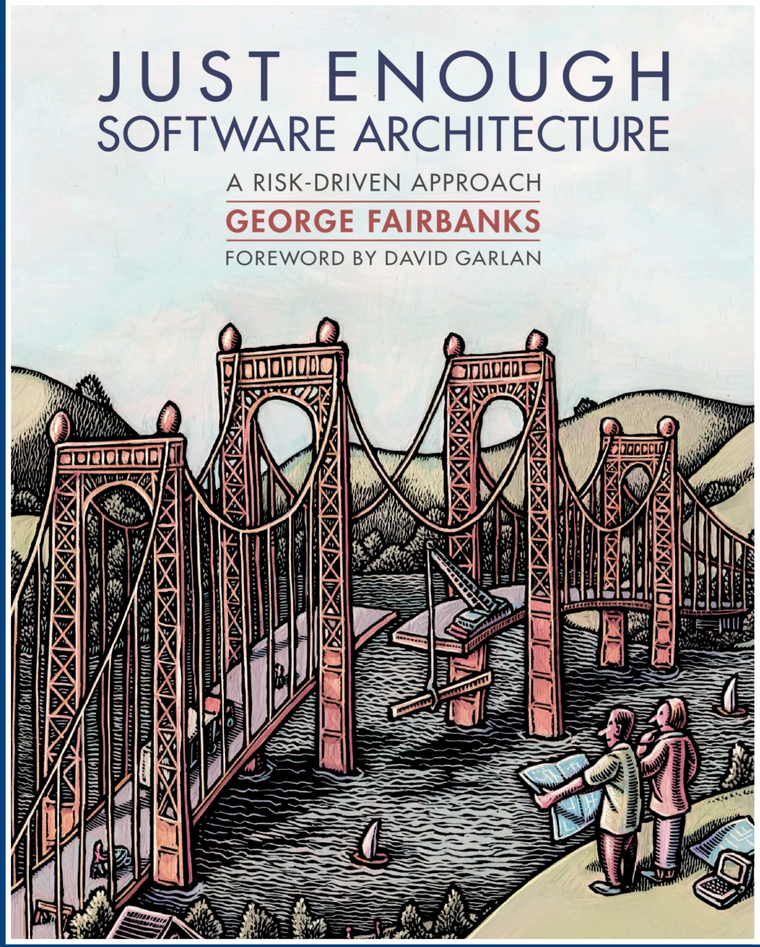
that help software developers navigate a large and/or complex codebase



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

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.

SIGSOFT '95 Washington, D.C., USA

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].

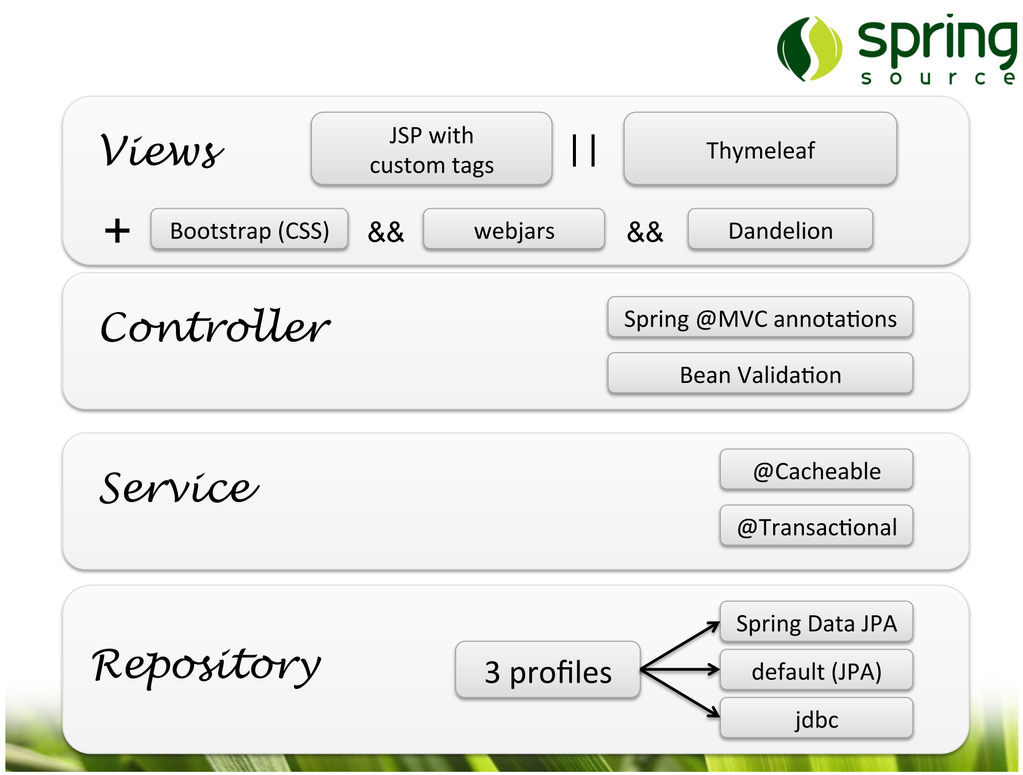
Would you code it that way?

(ensure that your diagrams reflect
your implementation intent)

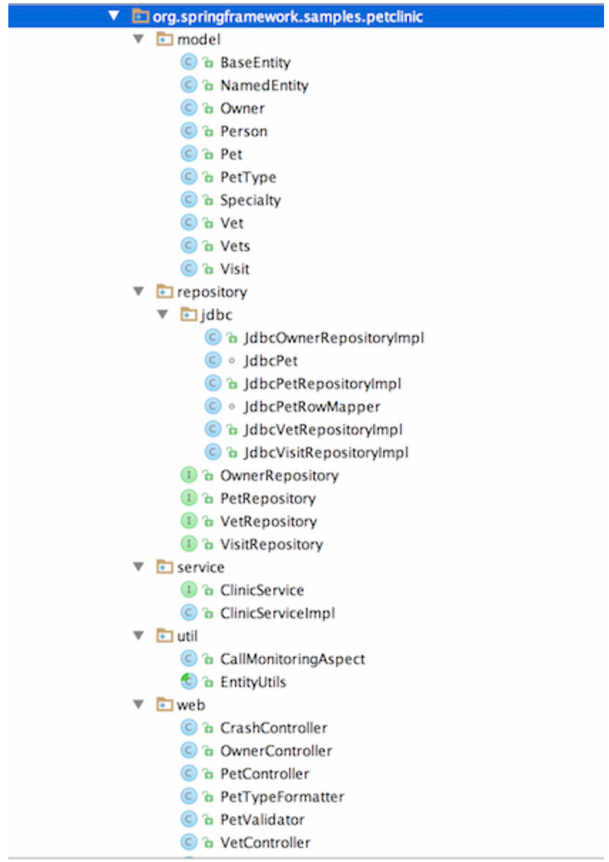
Spring PetClinic

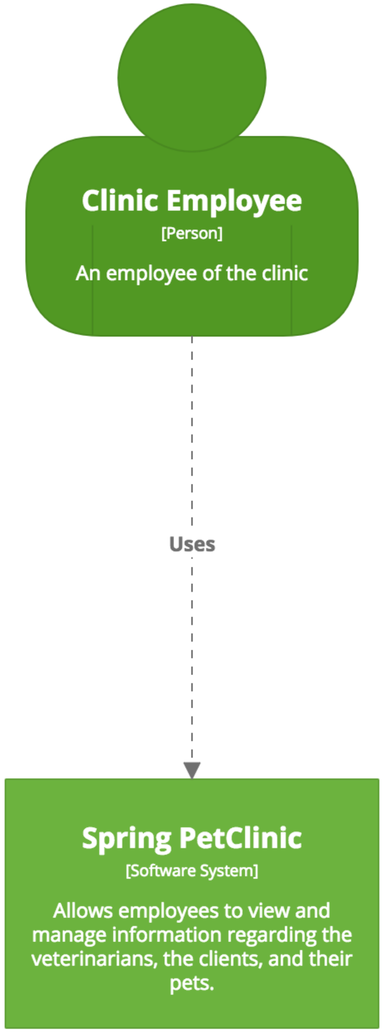
A sample application that illustrates how to build Java web applications using the Spring MVC framework

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



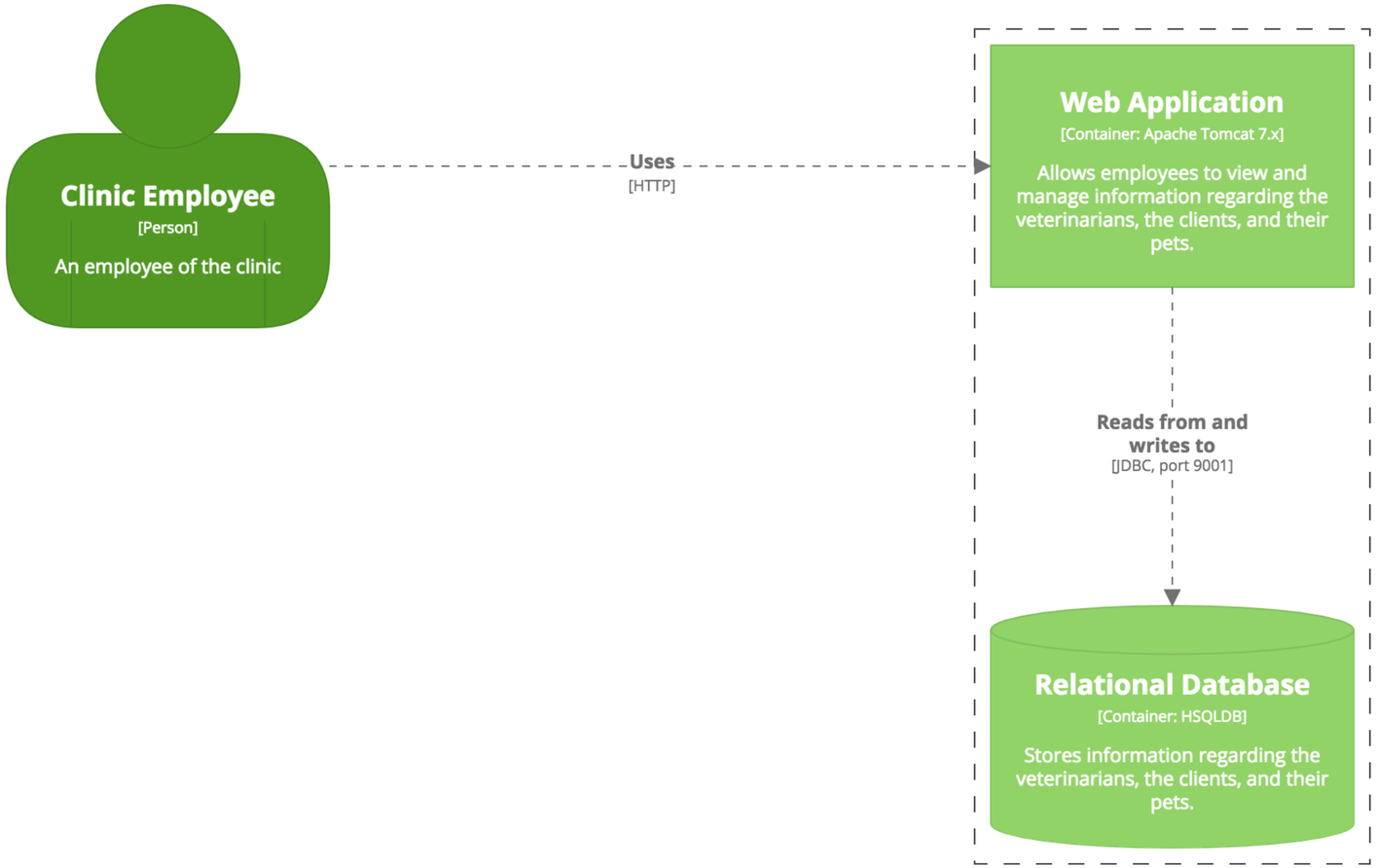
https://speakerdeck.com/michaelisvy/spring-petclinic-sample-application





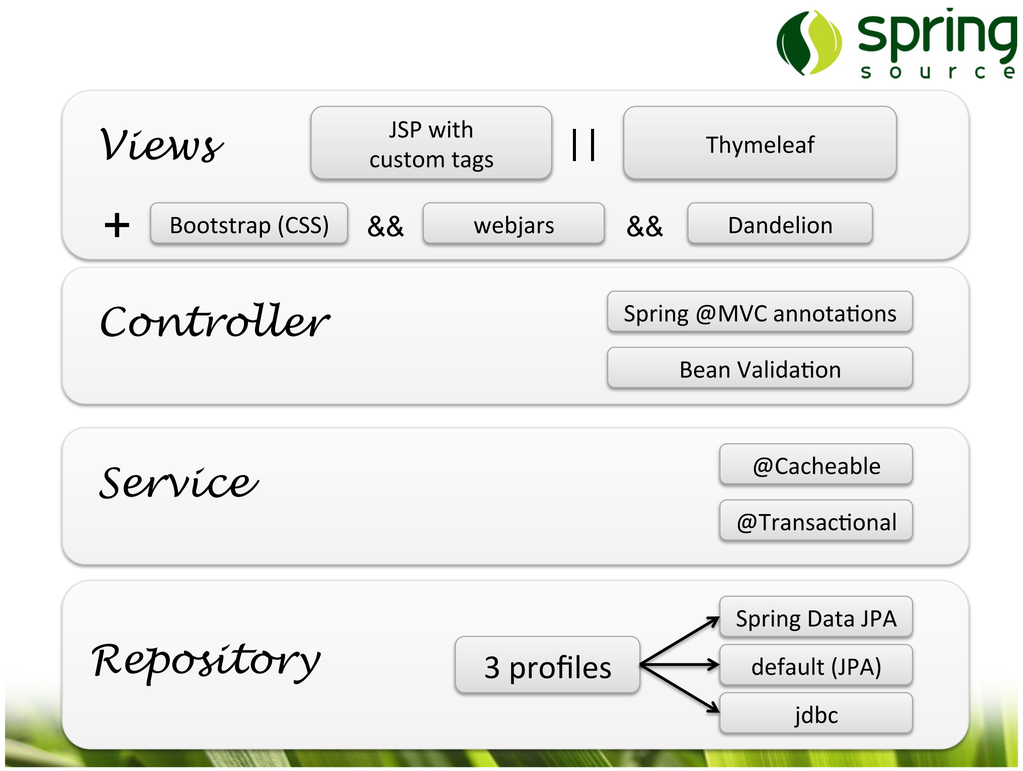
System Context diagram for Spring PetClinic

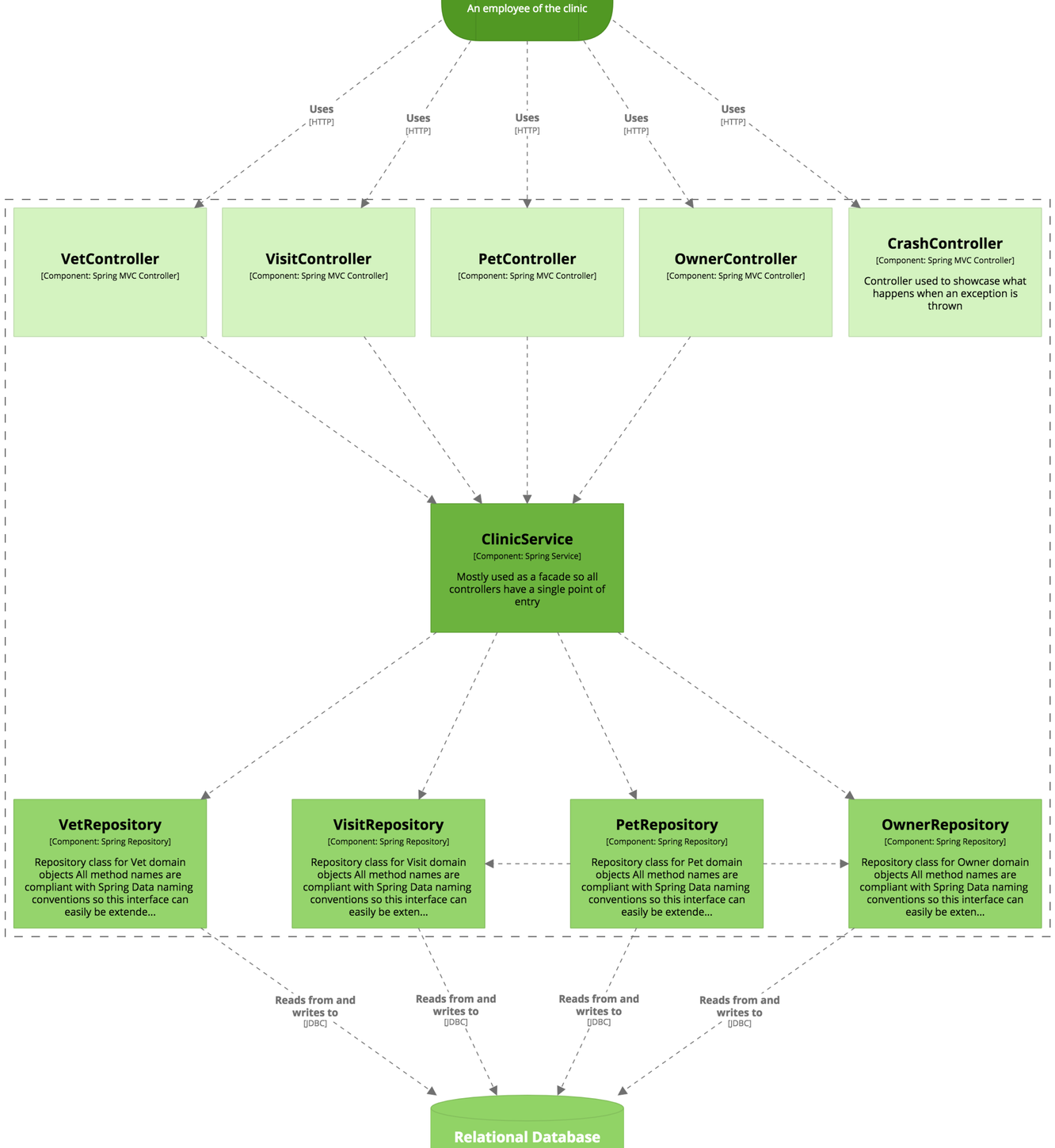
The System Context diagram for the Spring PetClinic system.
Friday 18 November 2016 22:21 UTC

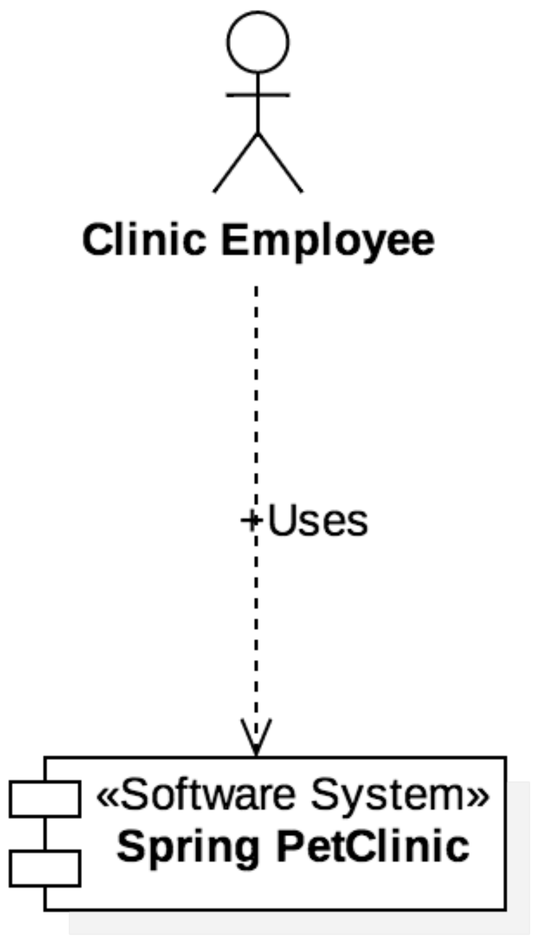


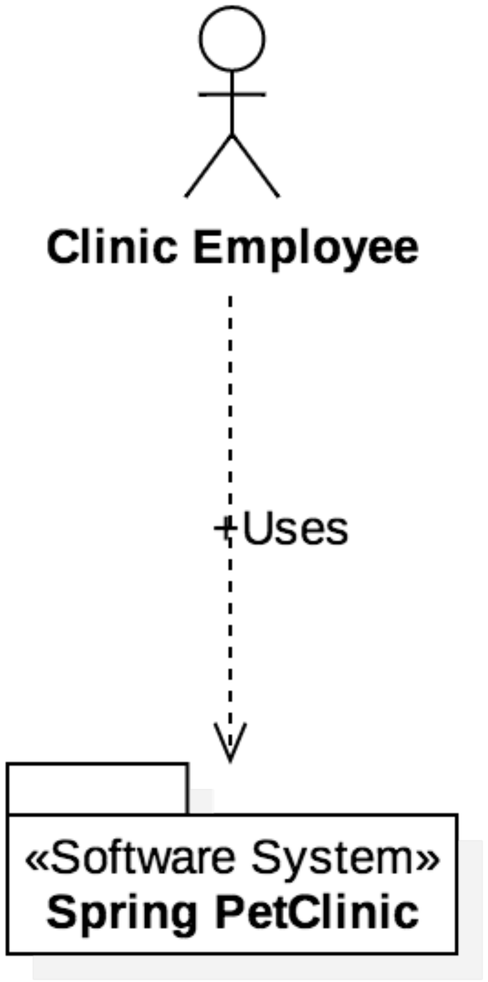
Container diagram for Spring PetClinic

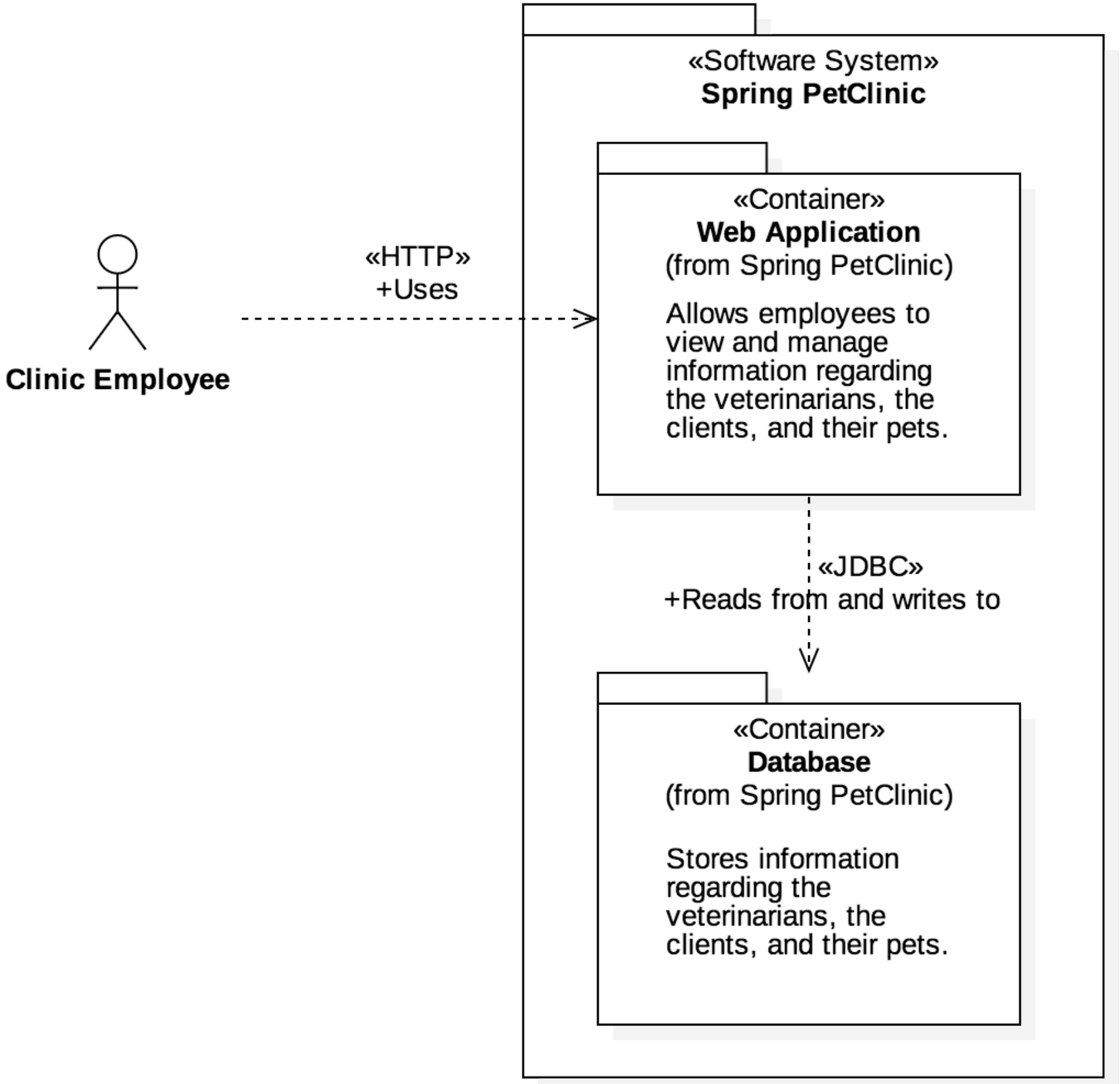
The Containers diagram for the Spring PetClinic system.
Friday 18 November 2016 22:21 UTC

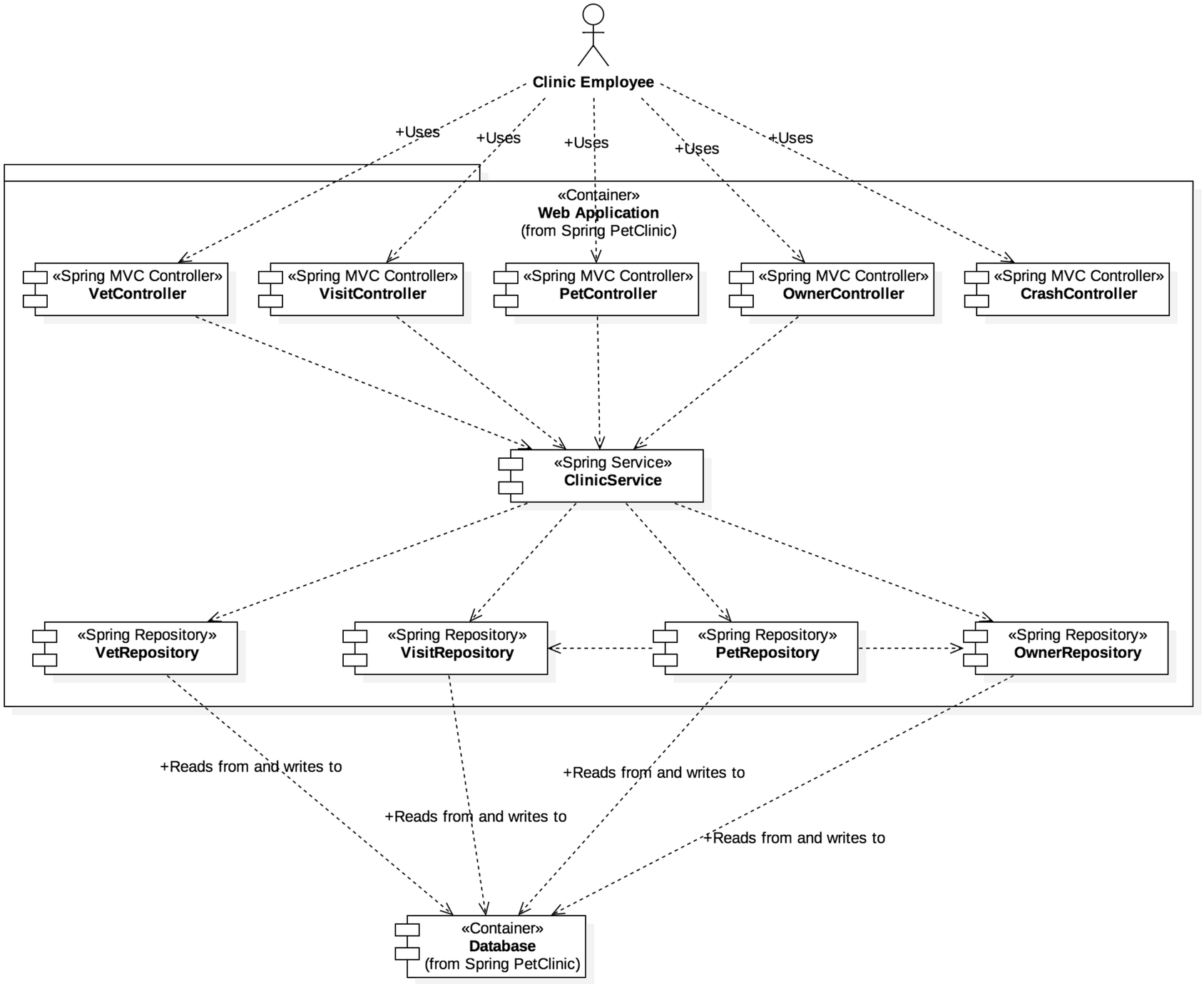

spring
source
Views
JSP with custom tags
||
Thymeleaf
+
Bootstrap (CSS)
&&
webjars
&&
Dandelion
Controller
Spring @MVC annotations
Bean Validation
Service
@Cacheable
@Transactional
Repository
3 profiles
Spring Data JPA
default (JPA)
jdbc


An employee of the clinic
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]
Relational Database

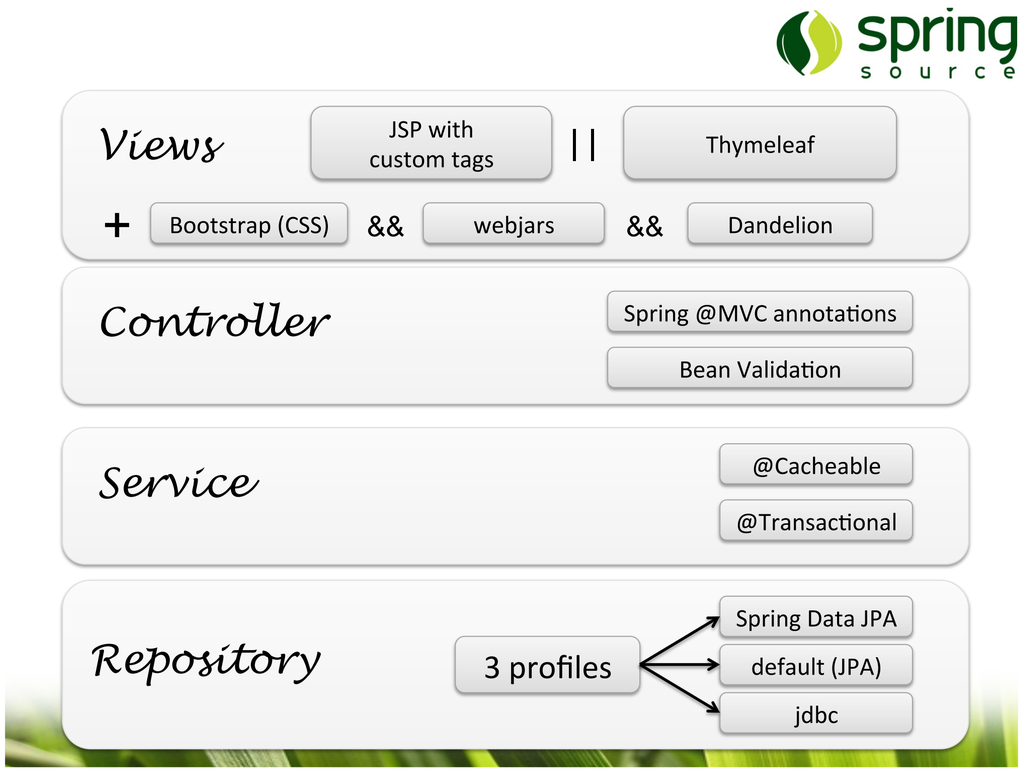

Clinic Employee
Uses
«Software System»
Spring PetClinic

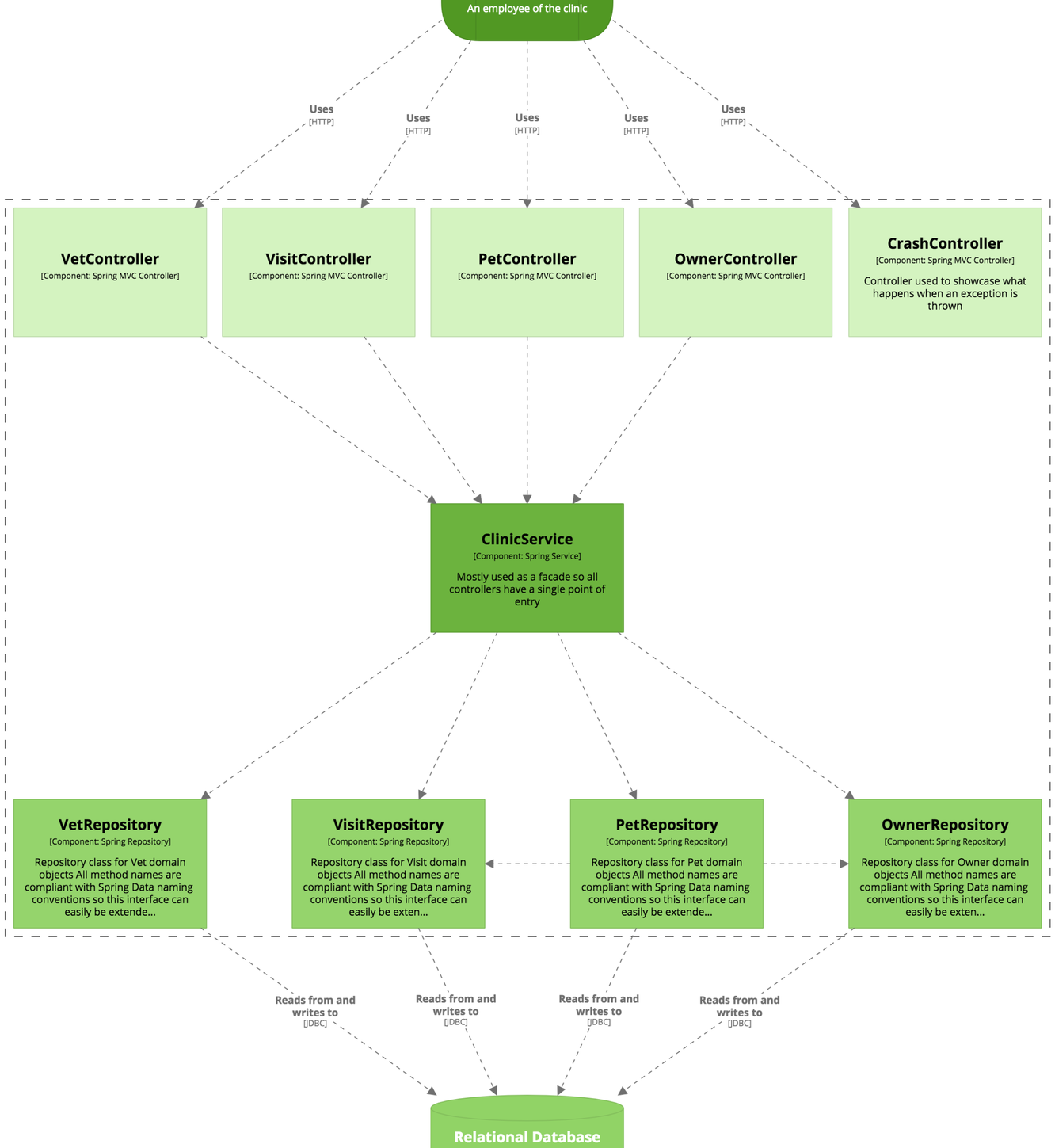

Clinic Employee
Uses
«Software System»
Spring PetClinic

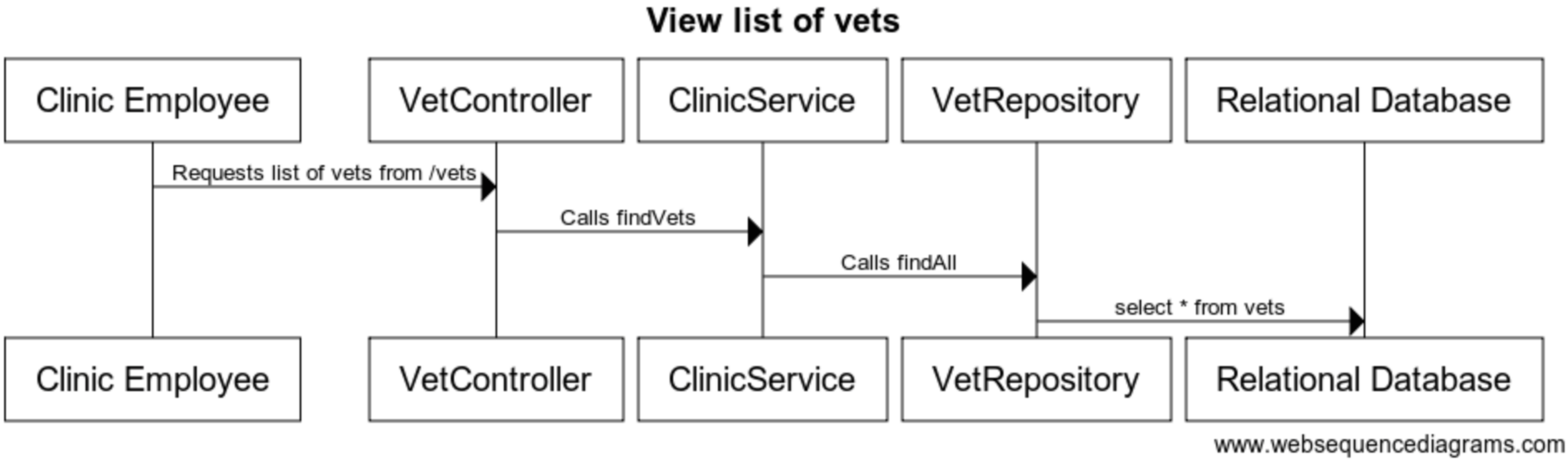

«Software System»
Spring PetClinic
«Container»
Web Application
(from Spring PetClinic)
Allows employees to view and manage information regarding the veterinarians, the clients, and their pets.
«HTTP»
+Uses
Clinic Employee
«JDBC»
+Reads from and writes to
«Container»
Database
(from Spring PetClinic)
Stores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

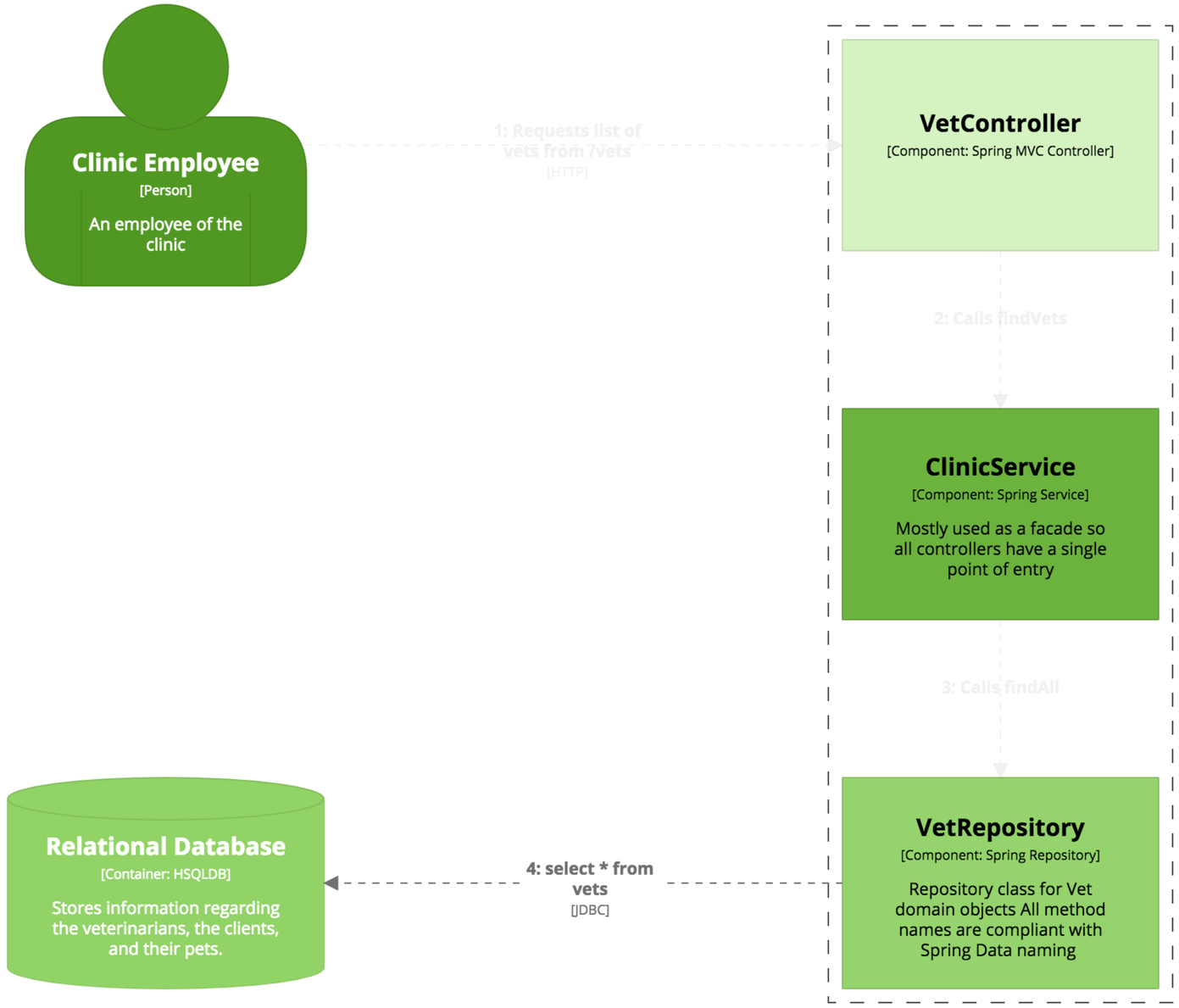

Clinic Employee
+Uses
+Uses
+Uses
+Uses
+Uses
«Container»
Web Application
(from Spring PetClinic)
«Spring MVC Controller»
VetController
«Spring MVC Controller»
VisitController
«Spring MVC Controller»
PetController
«Spring MVC Controller»
OwnerController
«Spring MVC Controller»
CrashController
«Spring 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)

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

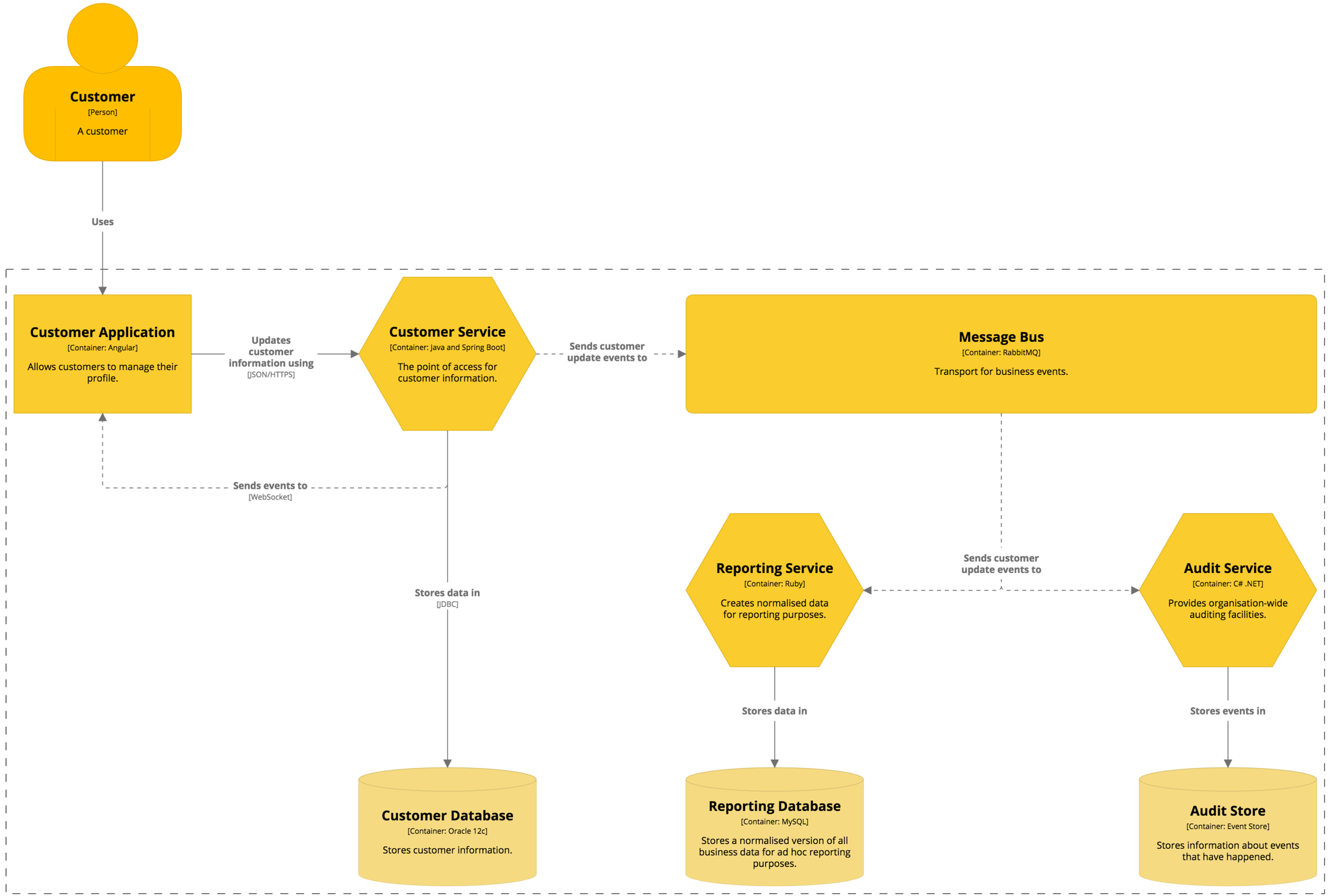

An employee of the clinic
Uses [HTTP]
Uses [HTTP]
Uses [HTTP]
Uses [HTTP]
Uses [HTTP]
VetController
[Component: Spring MVC Controller]
VisitController
[Component: Spring MVC Controller]
PetController
[Component: Spring MVC Controller]
OwnerController
[Component: Spring MVC Controller]
CrashController
[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]
Relational 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



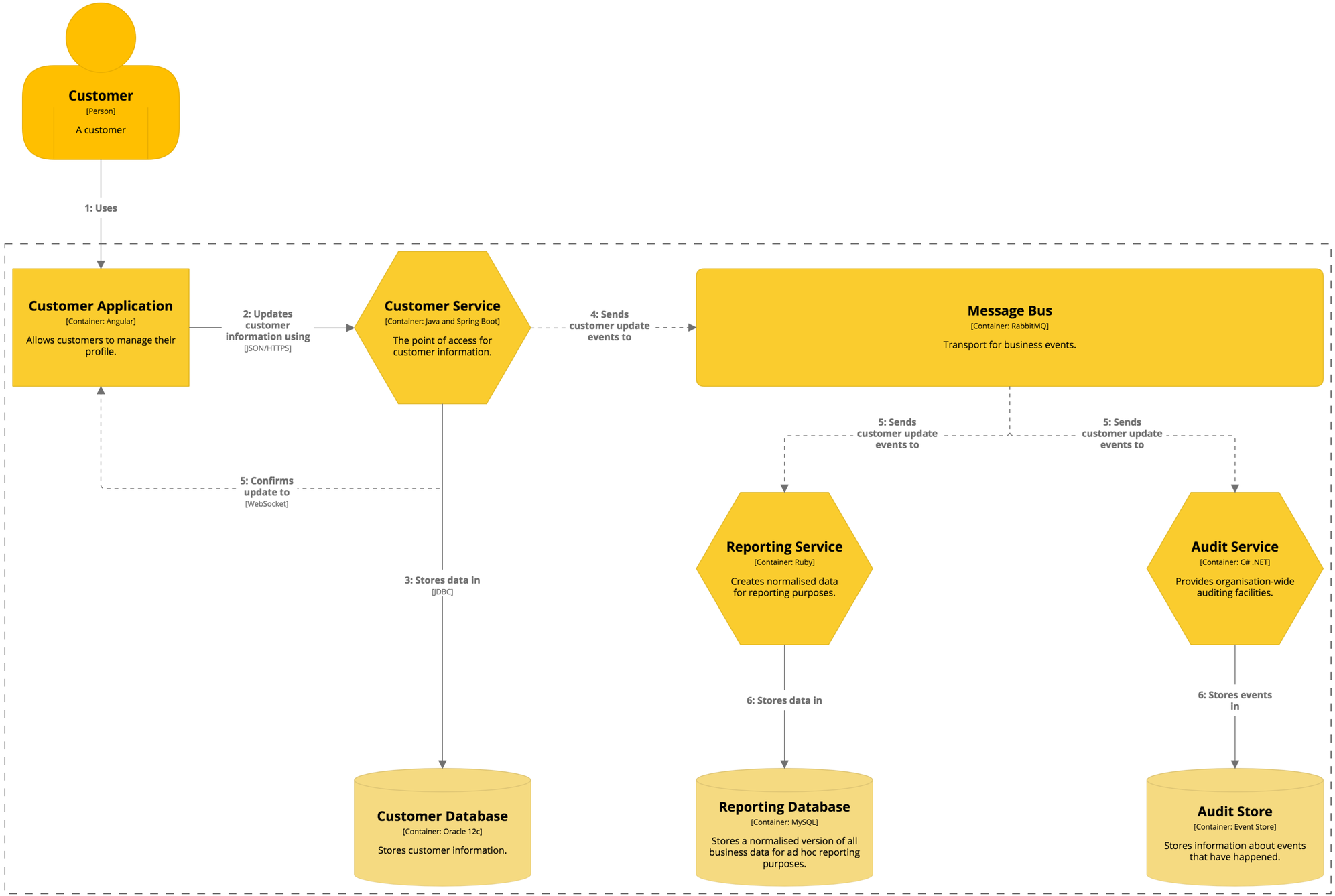
Dynamic diagram

Shows how the "view list of vets" feature works.
Sunday 20 November 2016 11:26 UTC



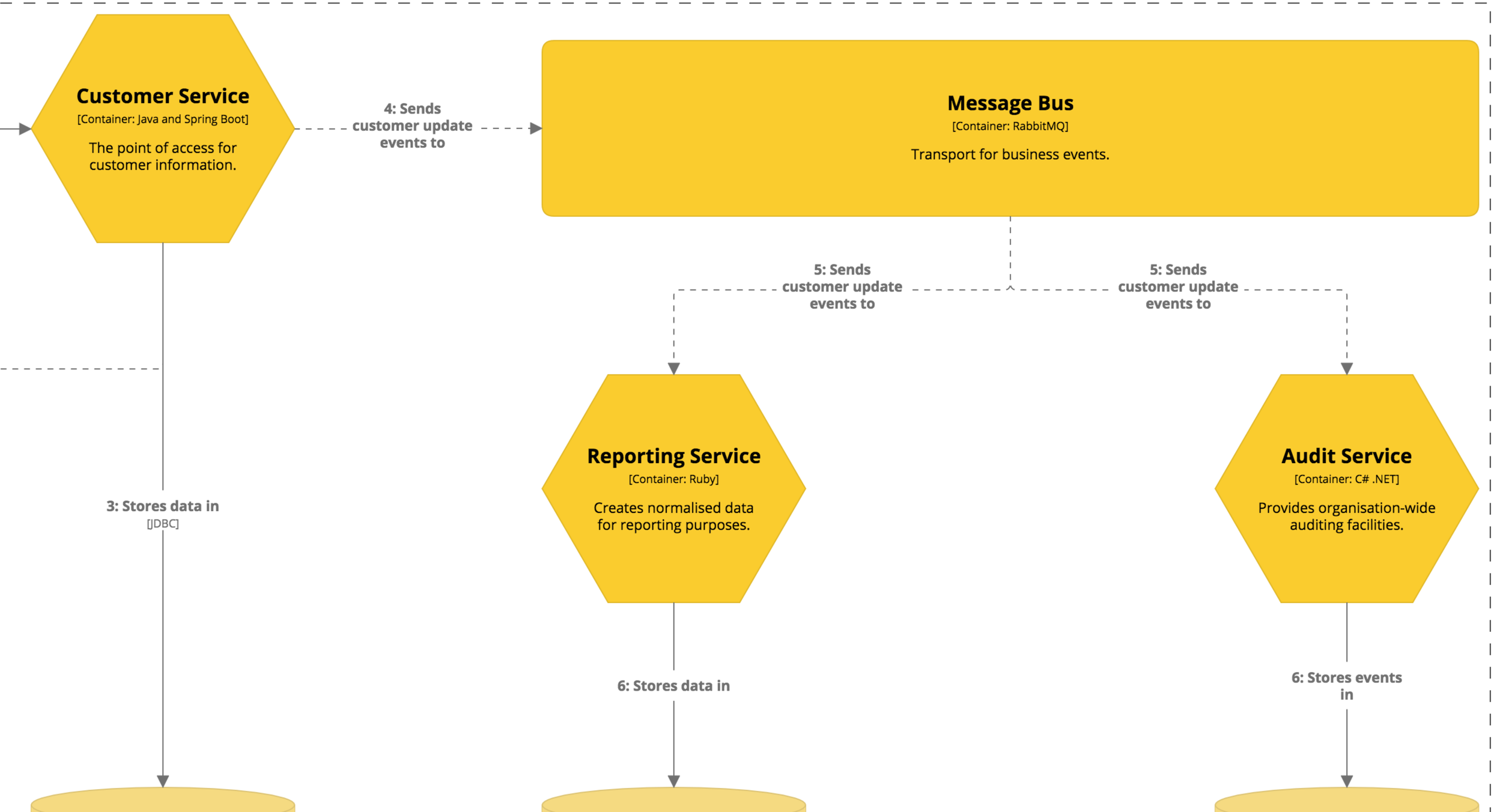
Container diagram for Customer Information System

Friday 04 November 2016 13:16 UTC



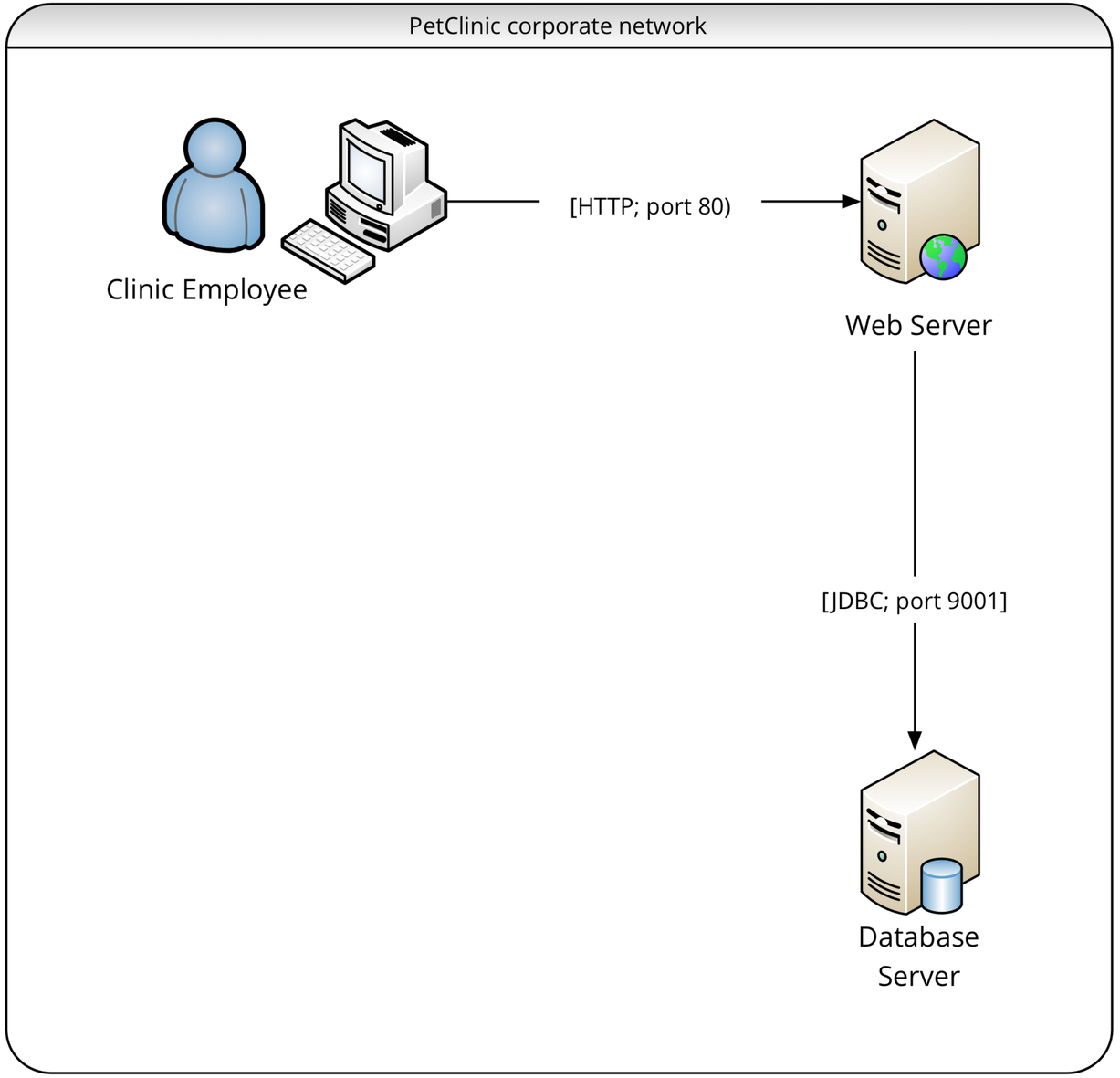
Dynamic diagram

This diagram shows what happens when a customer updates their details.
Friday 04 November 2016 13:16 UTC

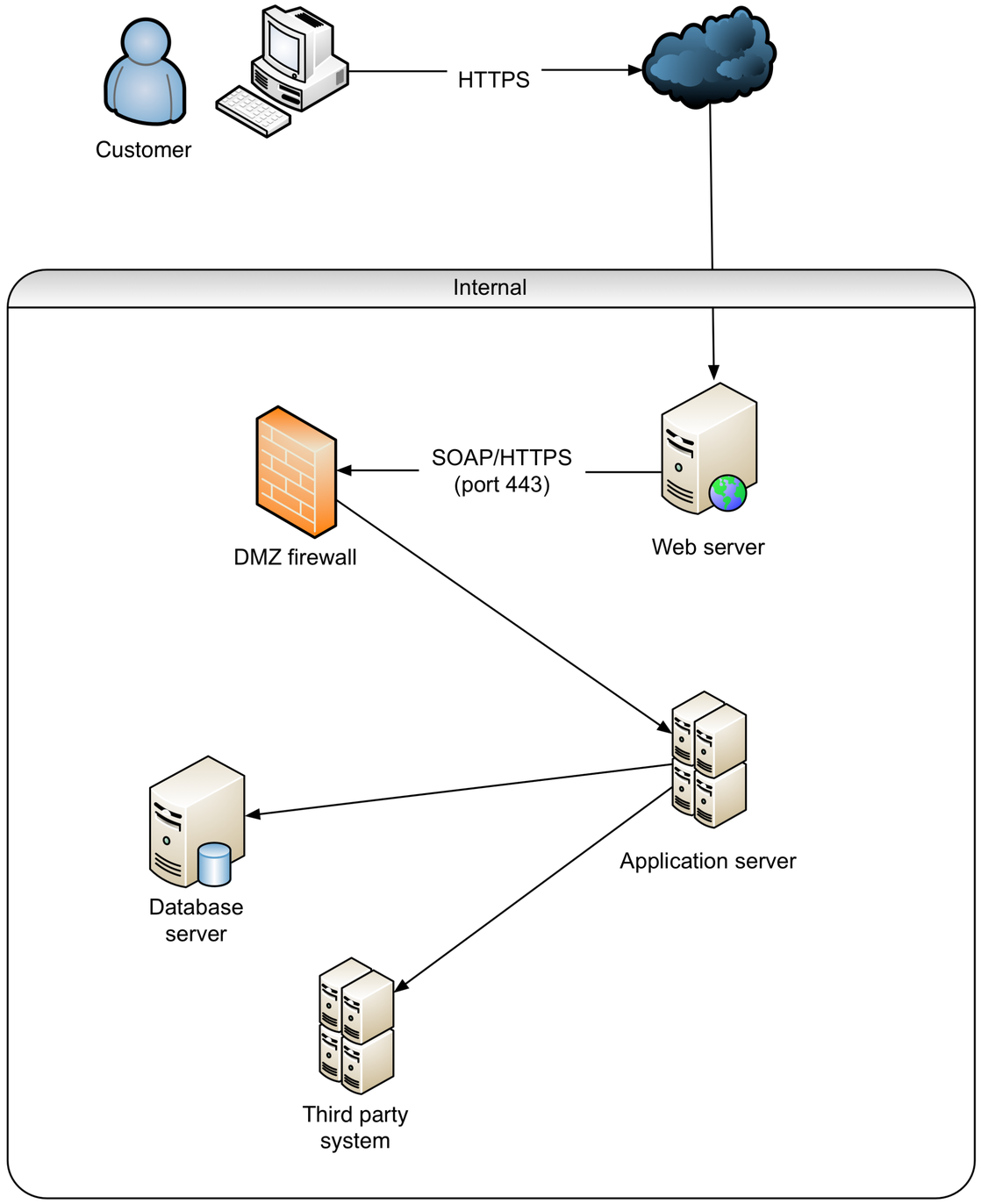

Customer Service
[Container: Java and Spring Boot]
The point of access for customer information.
4: Sends customer update events to
Message Bus
[Container: RabbitMQ]
Transport for business events.
5: Sends customer update events to
5: Sends customer update events to
Reporting Service
[Container: Ruby]
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

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

A map of the **infrastructure** can often be useful



[Infrastructure] Spring PetClinic

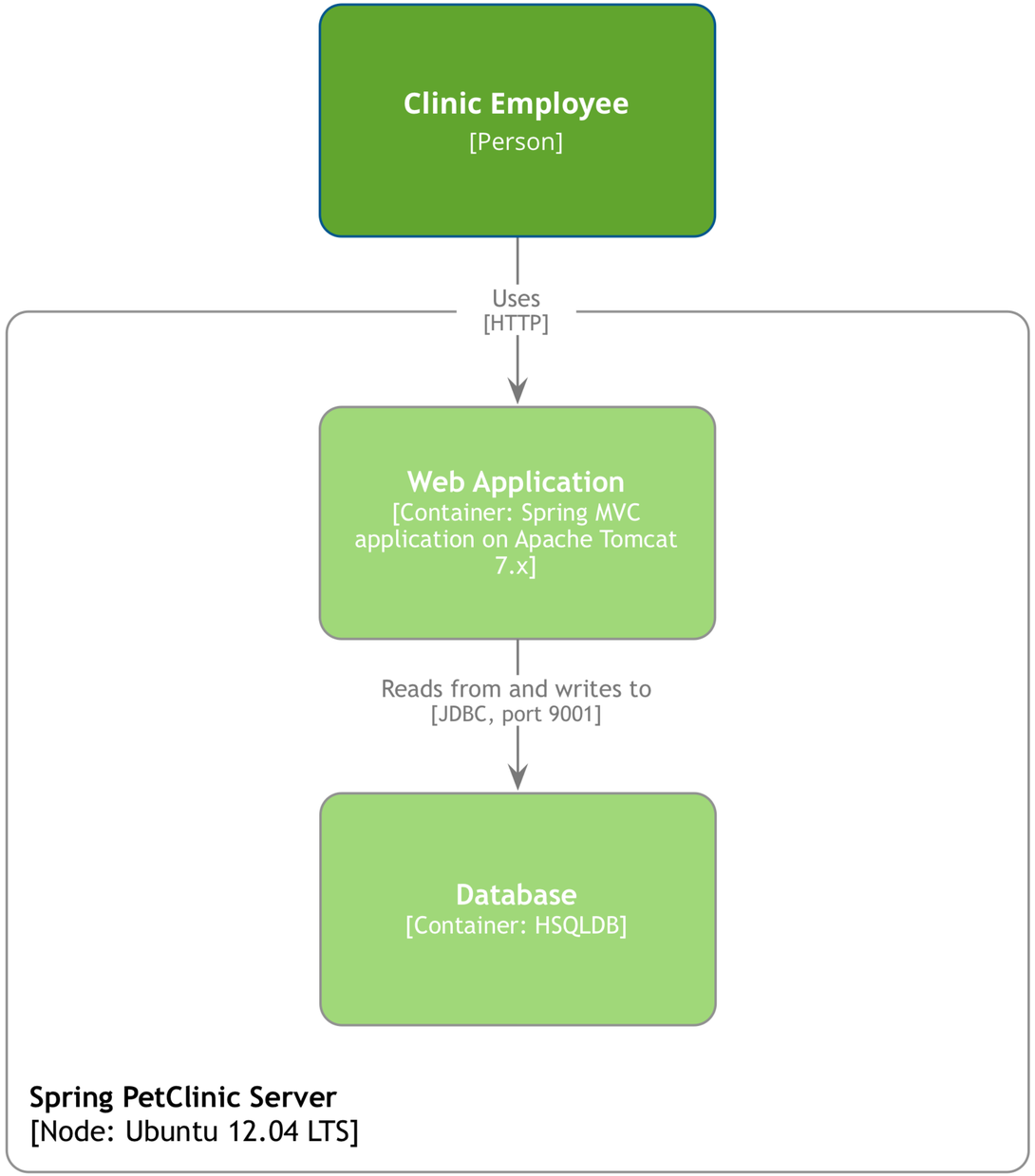

Customer
HTTPS
Internal
SOAP/HTTPS
(port 443)
DMZ firewall
Web server
Application server
Database
server
Third party
system

Deployment is about the mapping of containers to infrastructure

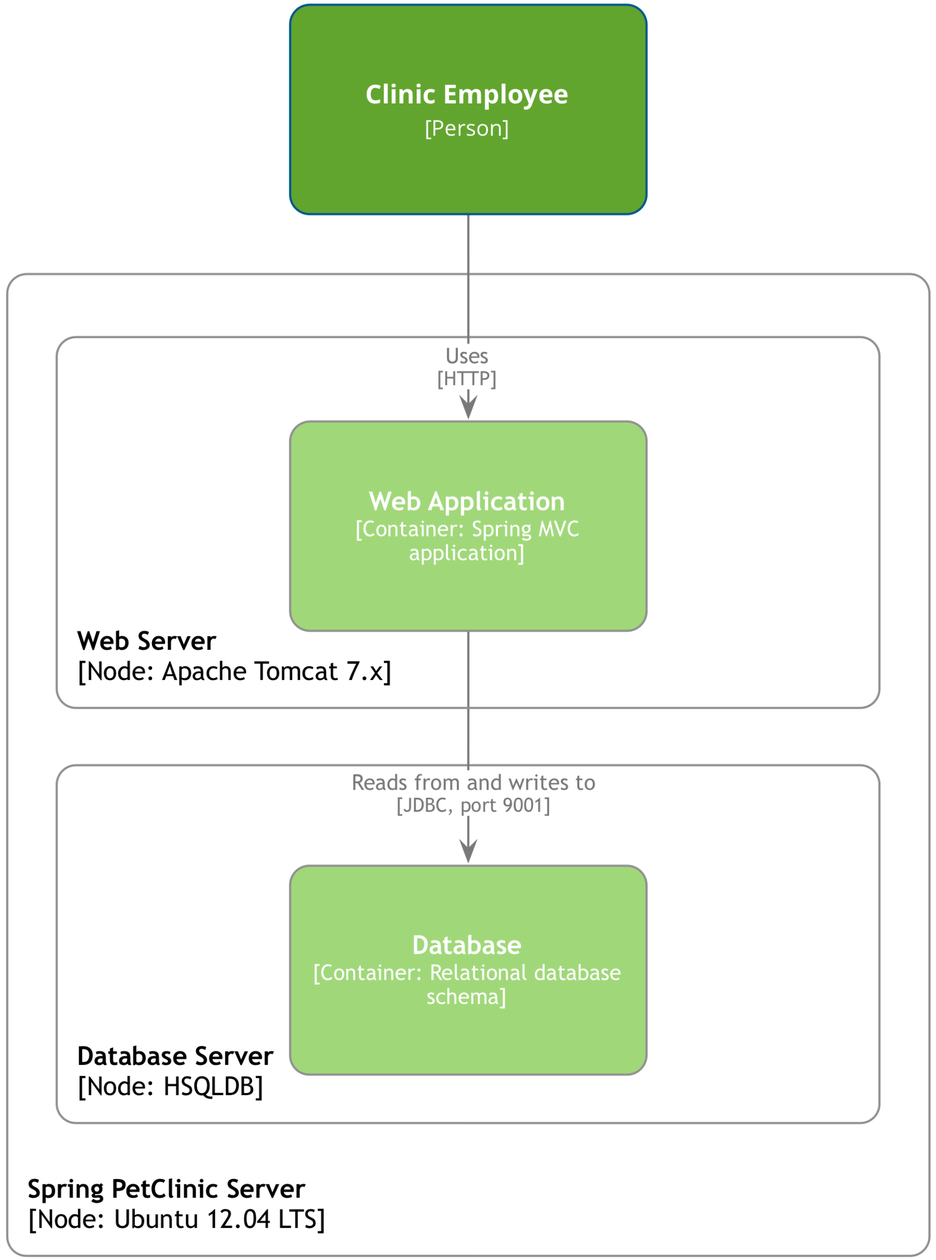
Deployment Node

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

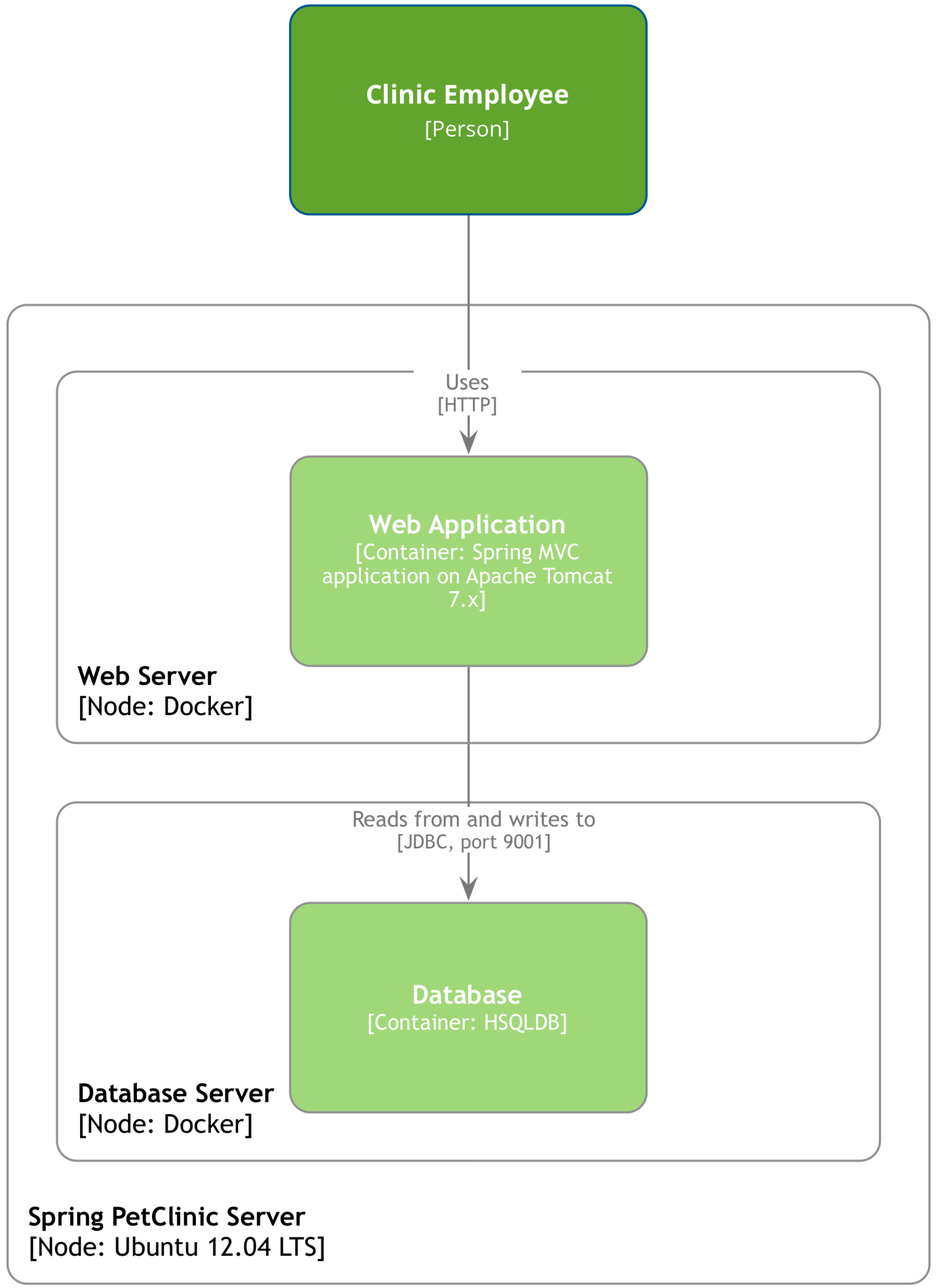
A deployment node can contain other deployment nodes or containers



[Deployment] Spring PetClinic



[Deployment] Spring PetClinic



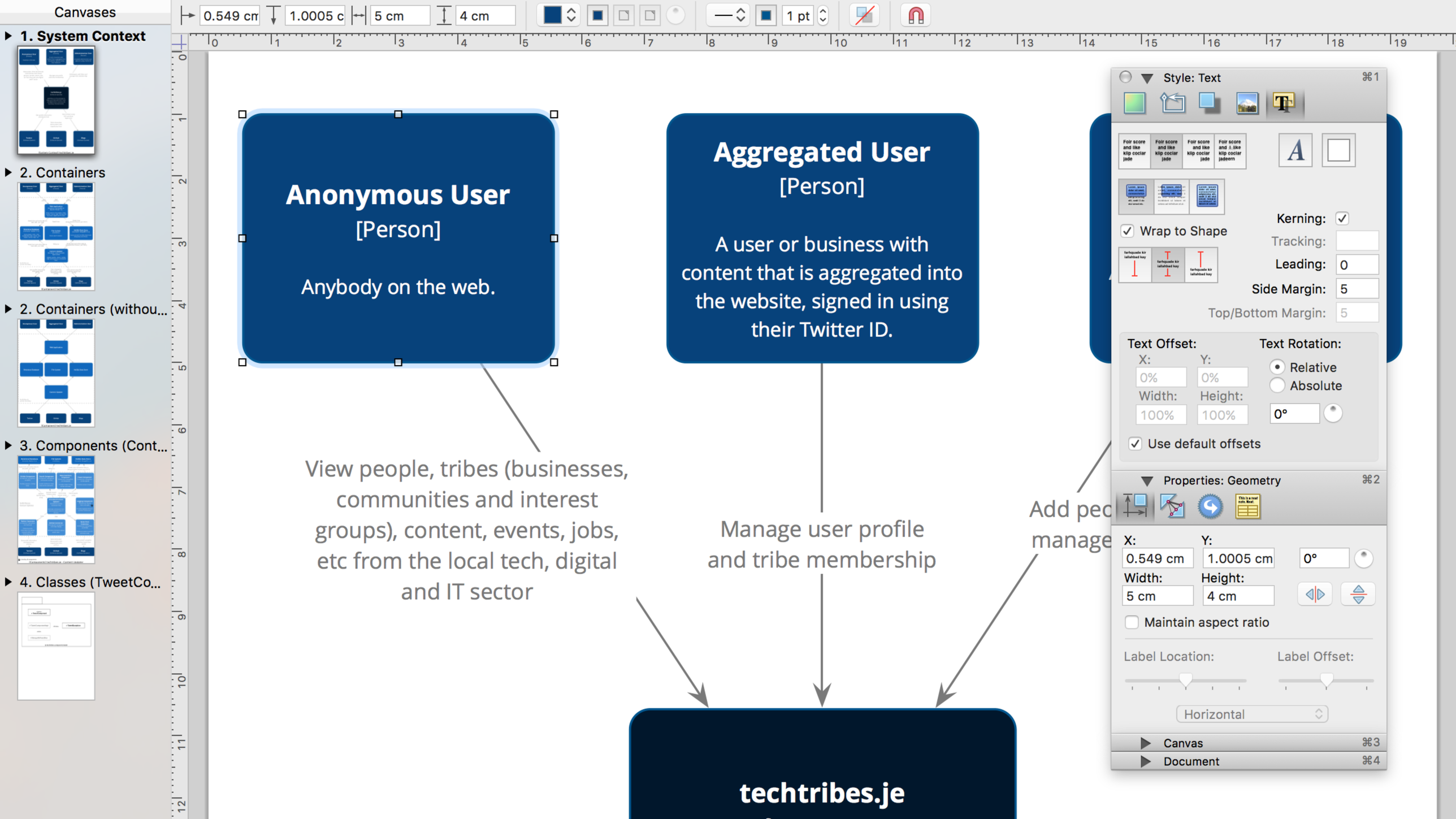
[Deployment] Spring PetClinic

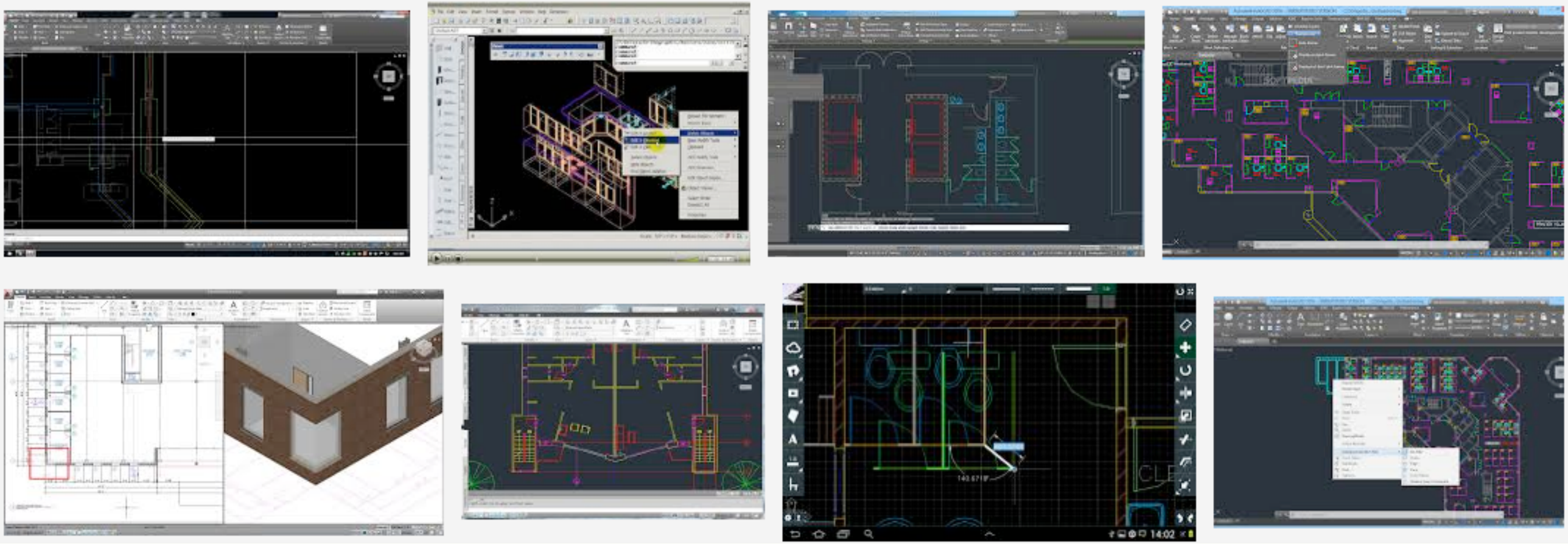
Tooling

What **tools** do you recommend?

Documentation format?

Microsoft Word, Microsoft SharePoint,
Atlassian Confluence, Markdown or AsciiDoc, etc


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
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



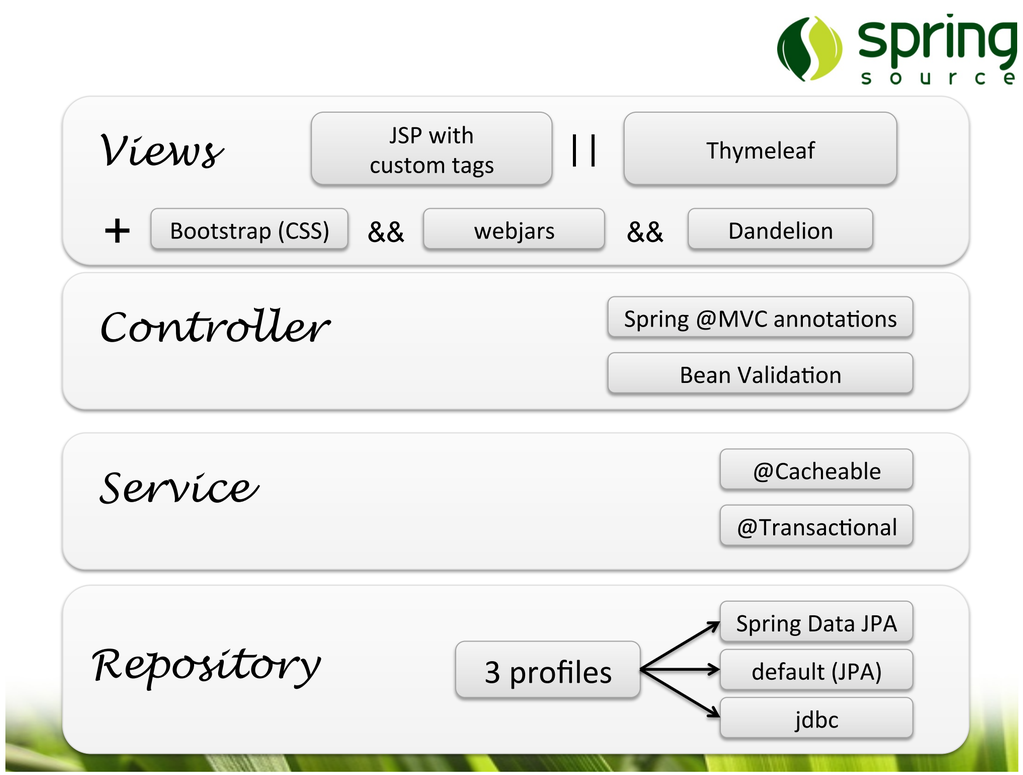
Do structural engineers and building architects use general purpose drawing tools?

Reverse-engineer
code to diagrams?

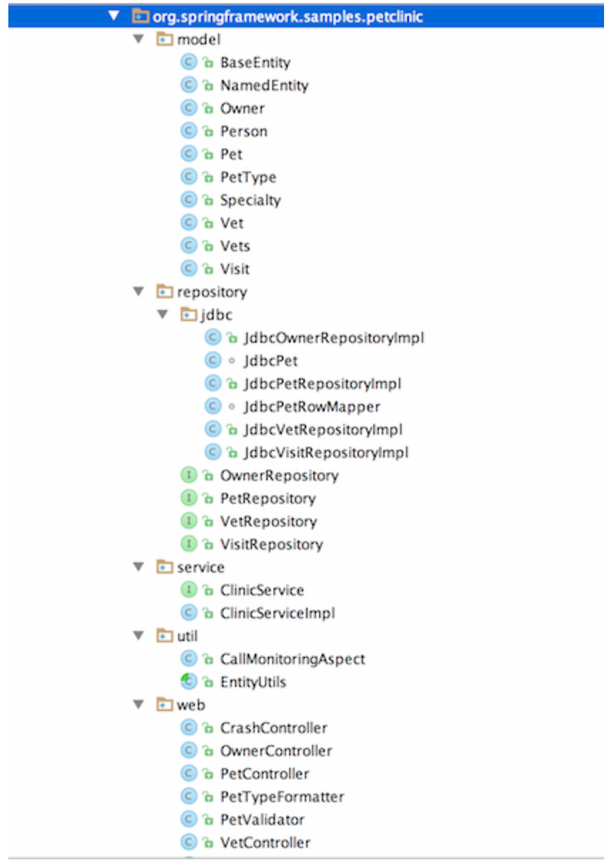
Spring PetClinic

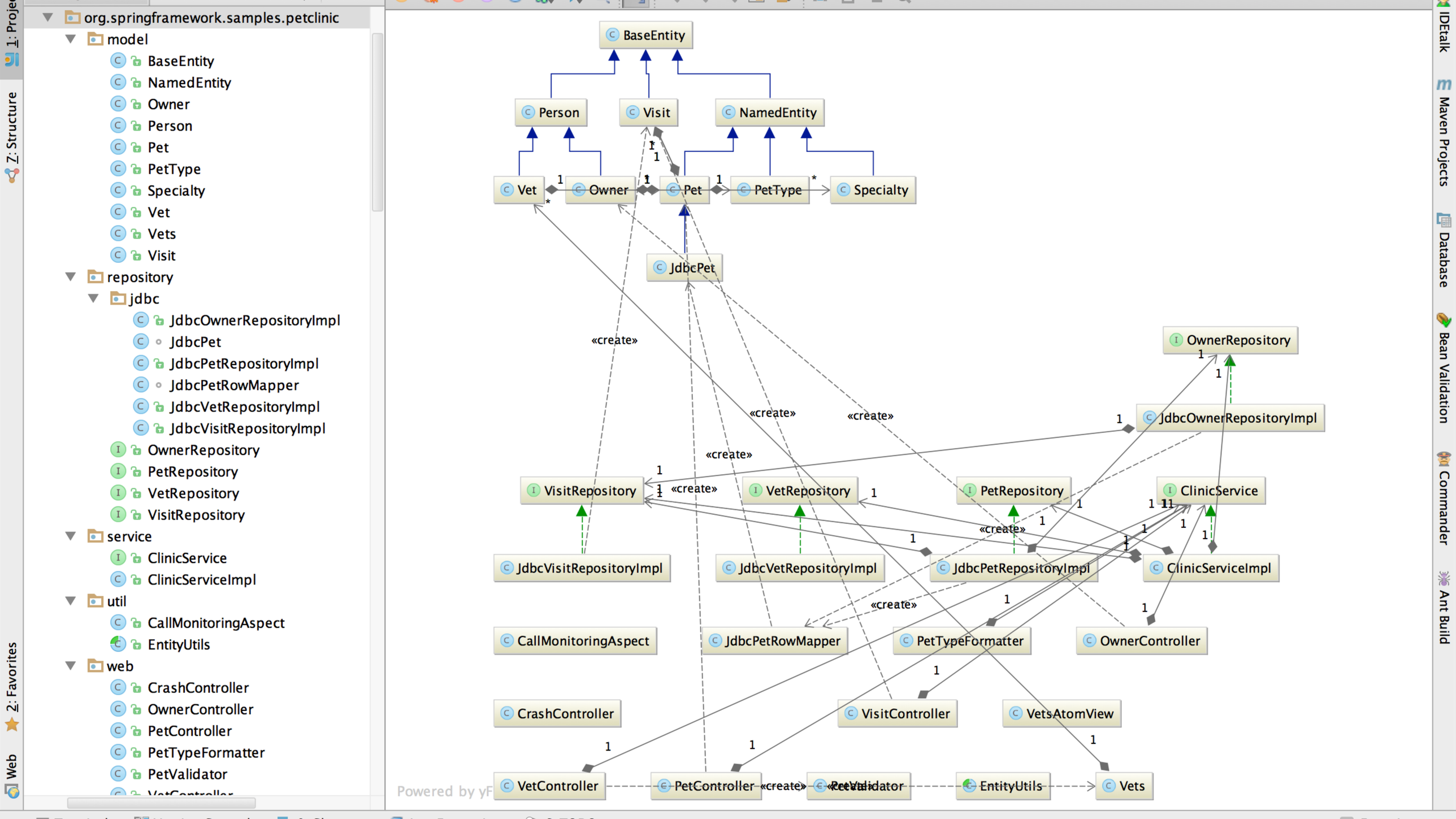
A sample application that illustrates how to build Java web applications using the Spring MVC framework

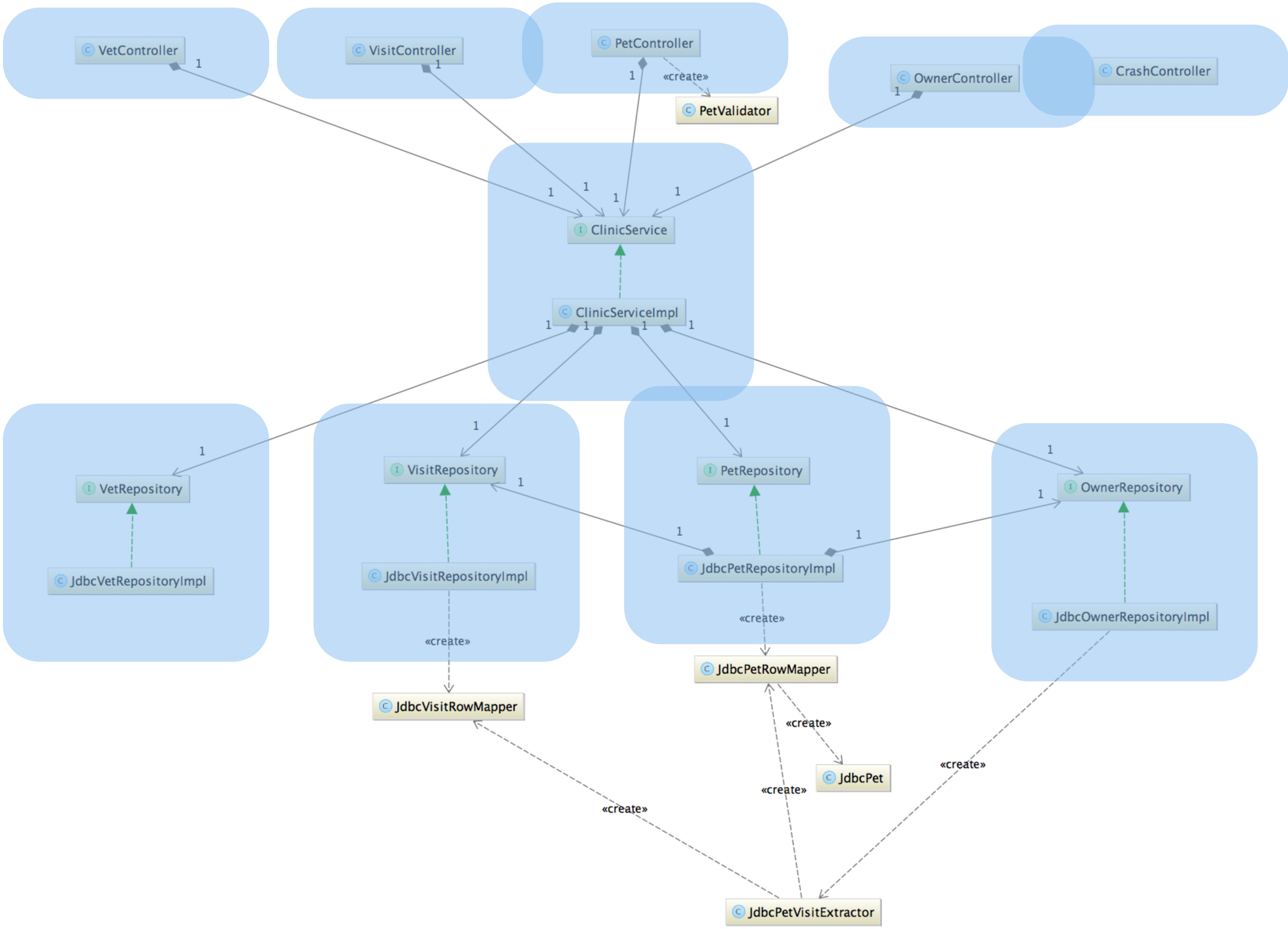
https://github.com/spring-projects/spring-petclinic/

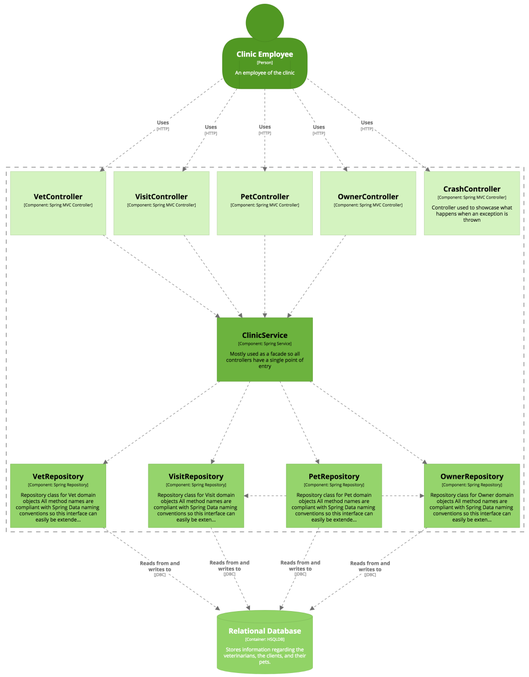


https://speakerdeck.com/michaelisvy/spring-petclinic-sample-application




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
1: Project
7: Structure
2: Favorites
Web
IDEtalk
Maven Projects
Database
Bean Validation
Commander
Ant Build


VetController
VisitController
PetController
«create»
PetValidator
OwnerController
CrashController
ClinicService
ClinicServiceImpl
VetRepository
JdbcVetRepositoryImpl
VisitRepository
JdbcVisitRepositoryImpl
PetRepository
JdbcPetRepositoryImpl
OwnerRepository
JdbcOwnerRepositoryImpl
JdbcPetRowMapper
JdbcVisitRowMapper
JdbcPet
JdbcPetVisitExtractor
1



Component diagram for Spring PetClinic - Web Application

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

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]

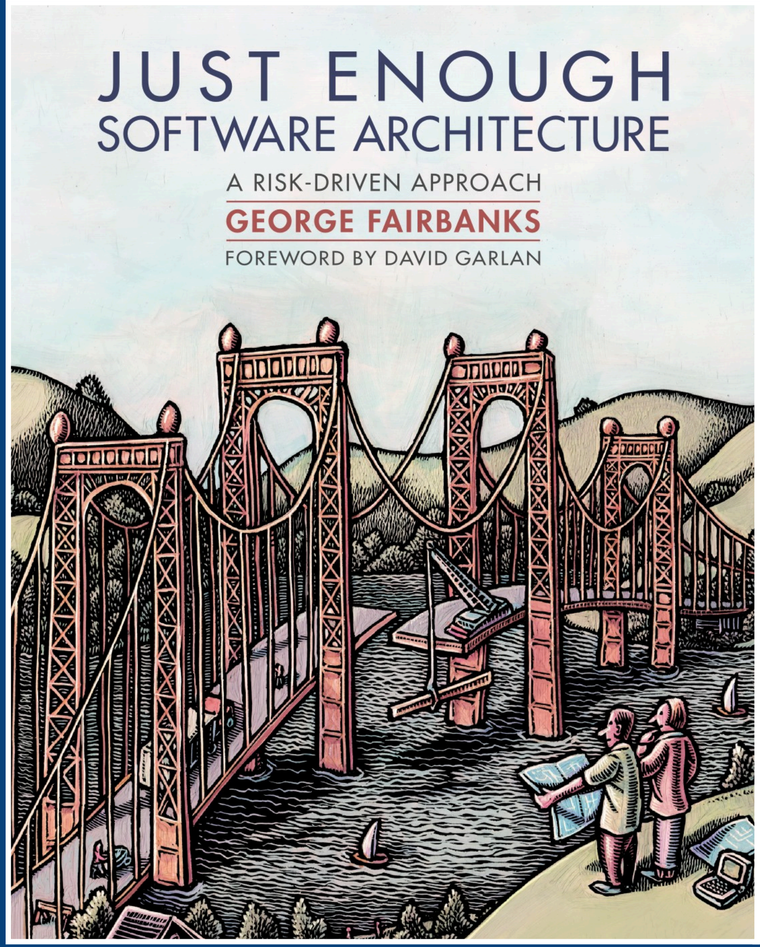
Repository class for Pet domain objects All method names are compliant with Spring Data naming conventions so this interface can easily be extende...

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...

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

Executable architecture description language

Structurizr for Java and .NET

```
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");
```

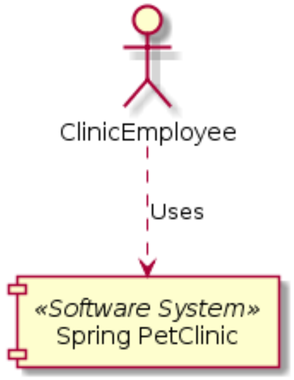
```
// components
ComponentFinder componentFinder = new ComponentFinder(
    webApplication,
    "org.springframework.samples.petclinic",
    new SpringComponentFinderStrategy(
        new ReferencedTypesSupportingTypesStrategy()
    ),
    new SourceCodeComponentFinderStrategy(
        new File(sourceRoot,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

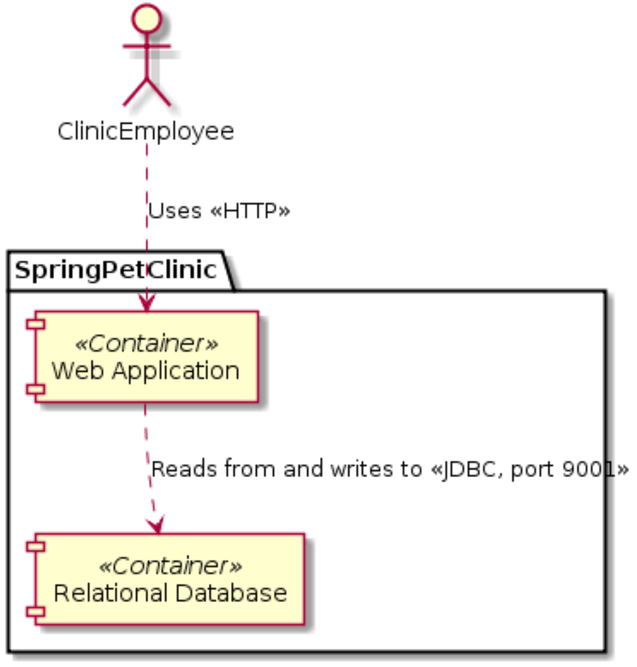
```
// system context, container and component views
ViewSet viewSet = workspace.getViews();

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

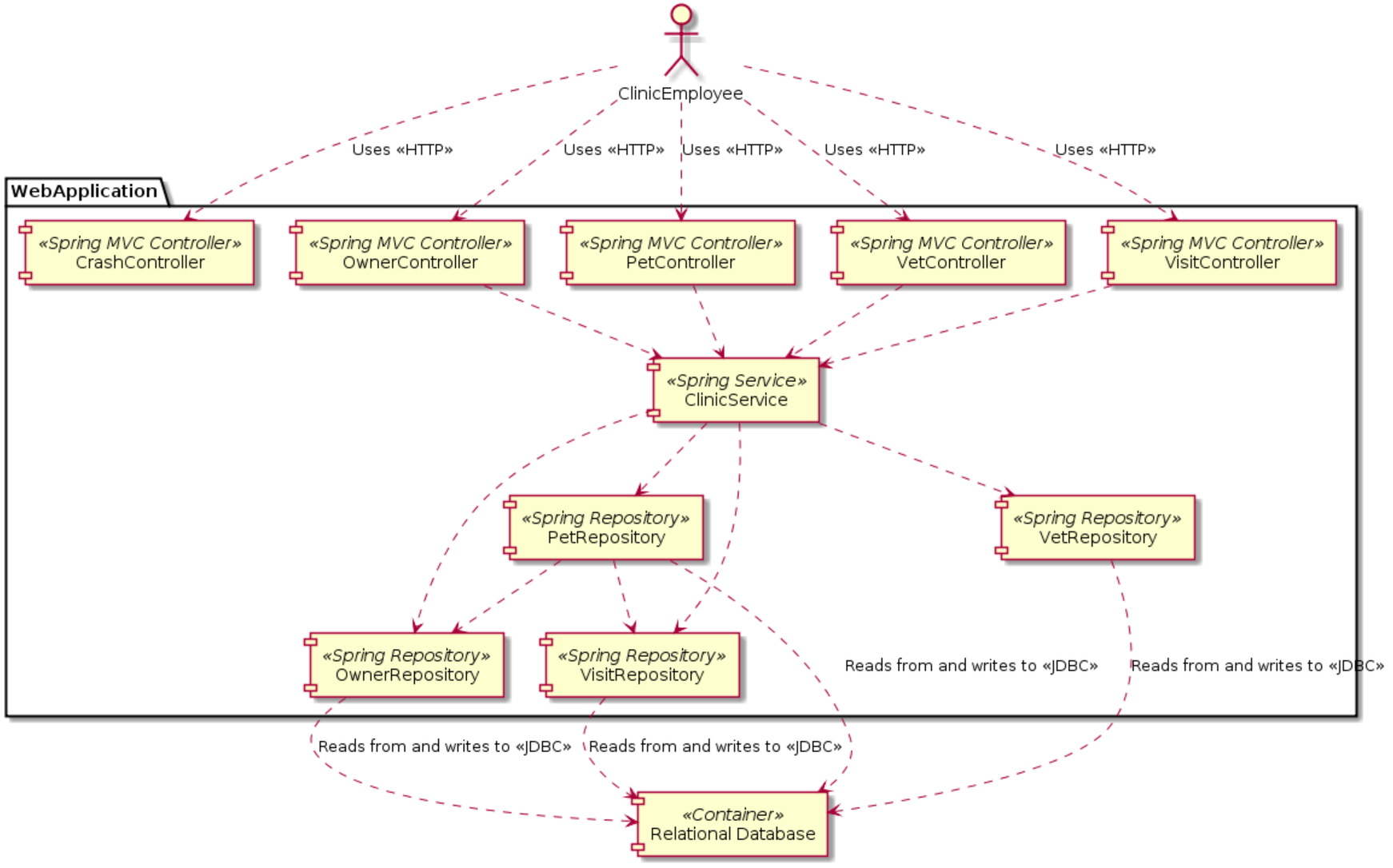
Spring PetClinic - System Context

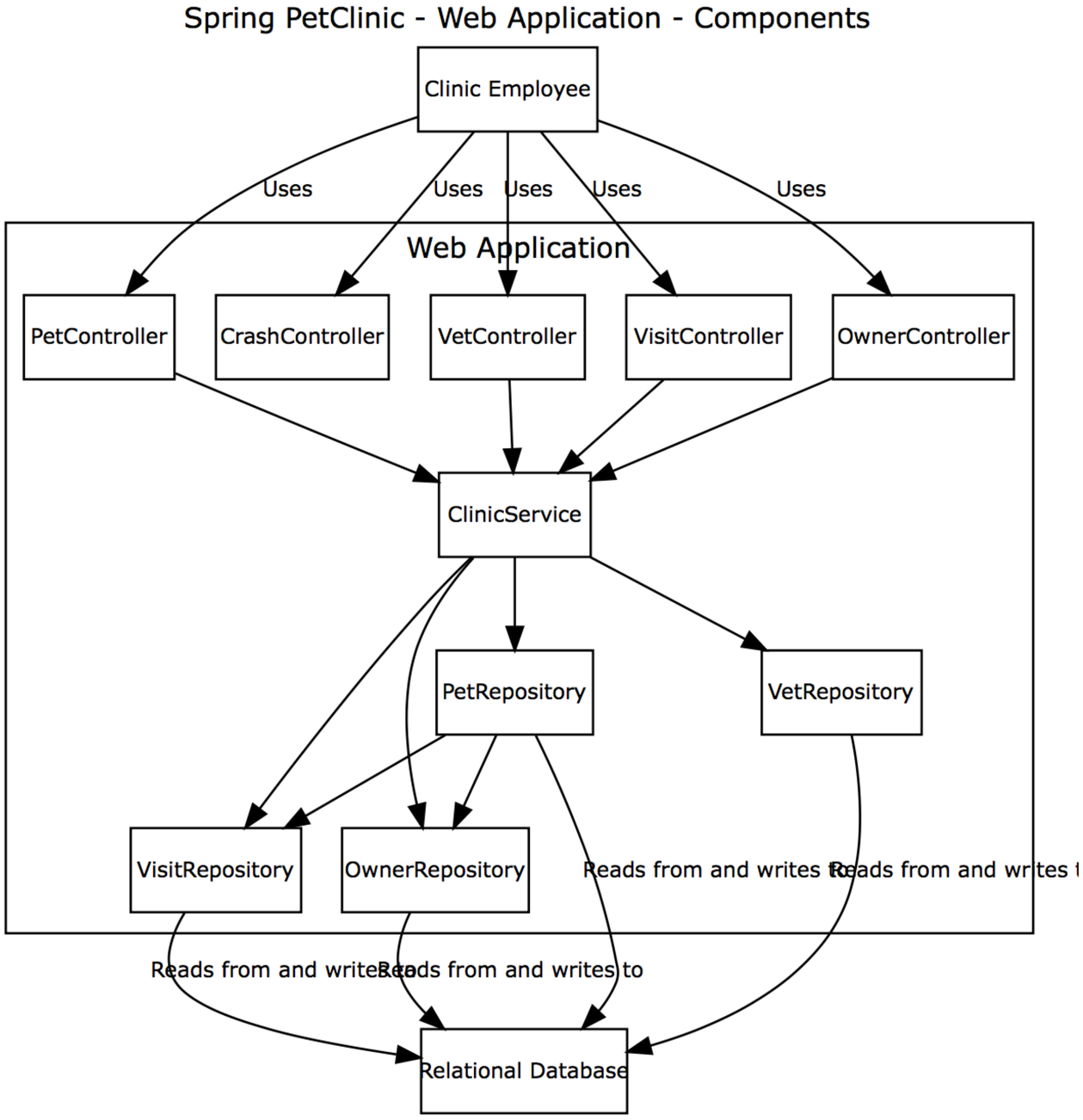


Spring PetClinic - Containers



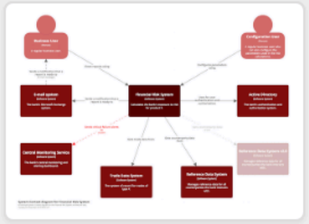
Spring PetClinic - Web Application - Components




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 to
Reads from and writes to
Reads from and writes to
Relational Database

Structurizr

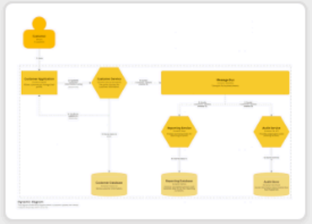
Visualise, document and explore your software architecture



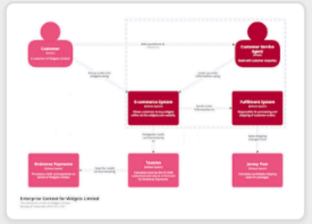
Create system context diagrams using Java code.

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

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

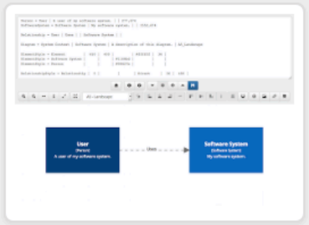


Create animated diagrams to describe dynamic behaviour.

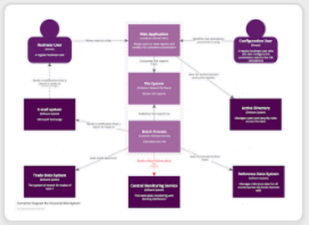


Create enterprise context diagrams using code.

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

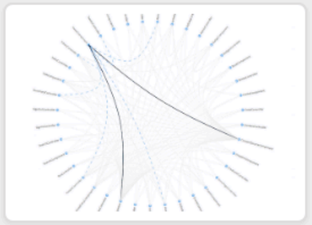


Create diagrams using text with Structurizr Express.

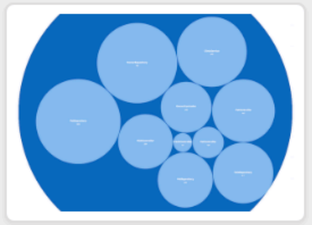


Create diagrams using C# code.

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



Explore inbound, outbound and cyclic dependencies between components.

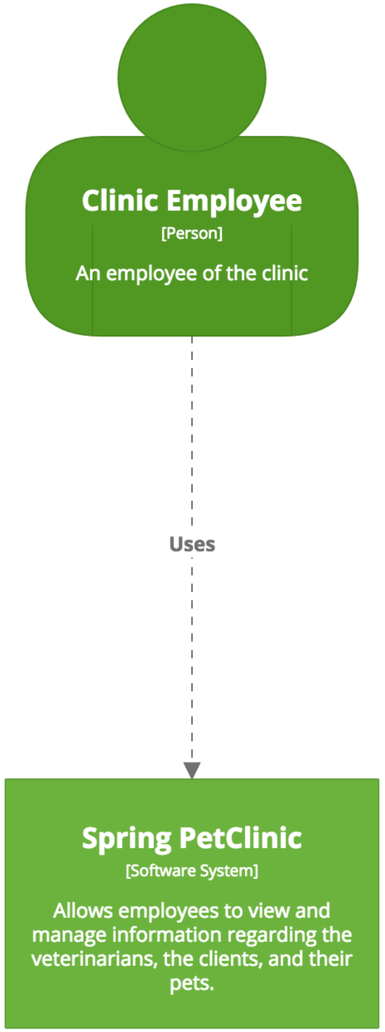


Explore component size.

Embed diagrams into Atlassian Confluence with prebuilt macros.

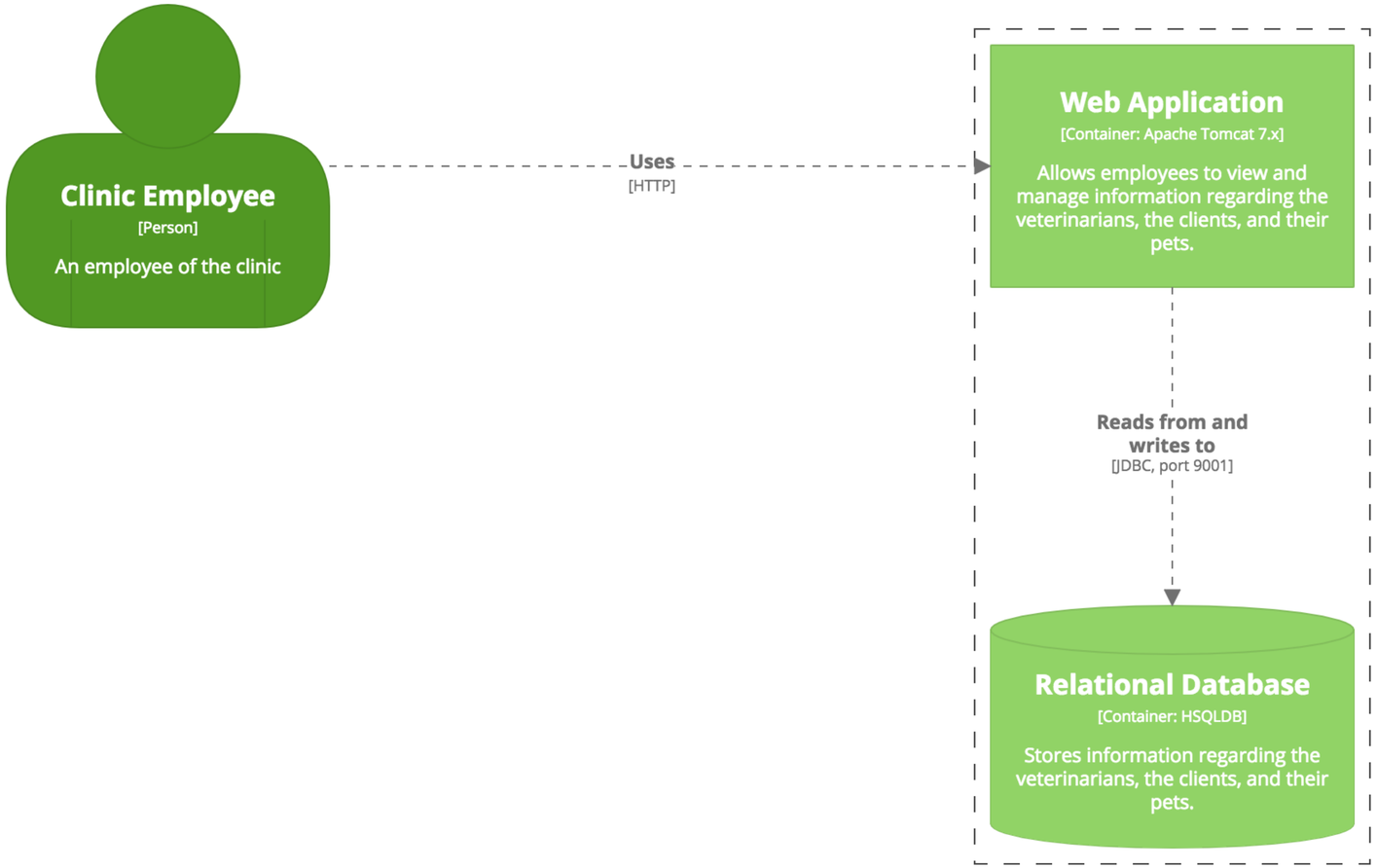
```
// upload the software architecture model to structurizr.com
StructurizrClient client = new StructurizrClient("key", "secret");
client.putWorkspace(1234, workspace);
```


```
{
 "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"
```



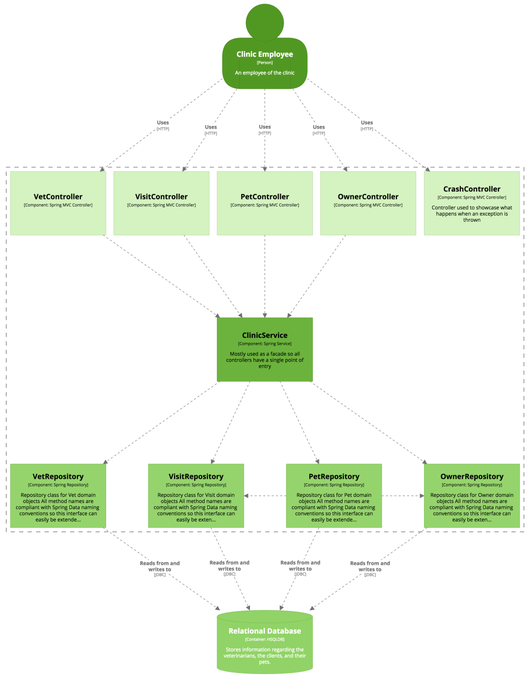
System Context diagram for Spring PetClinic

The System Context diagram for the Spring PetClinic system.
Friday 18 November 2016 22:21 UTC



Container diagram for Spring PetClinic

The Containers diagram for the Spring PetClinic system.
Friday 18 November 2016 22:21 UTC



Component diagram for Spring PetClinic - Web Application

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

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]

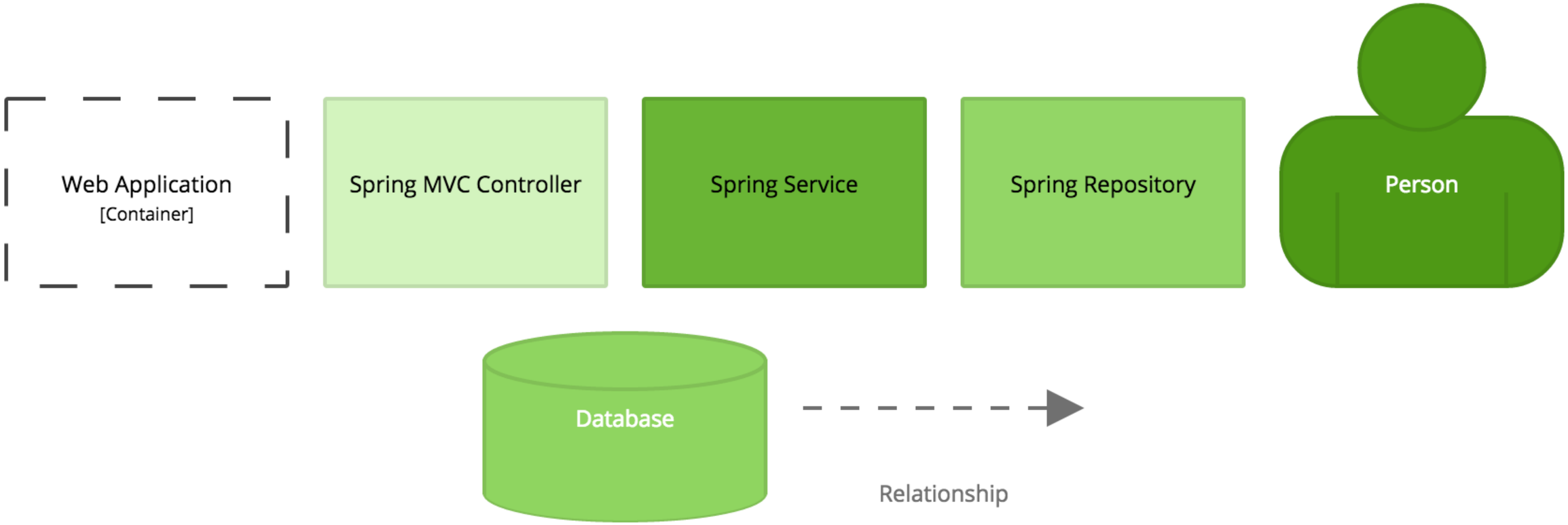
Repository class for Pet domain objects All method names are compliant with Spring Data naming conventions so this interface can easily be extende...

OwnerRepository
[Component: Spring Repository]

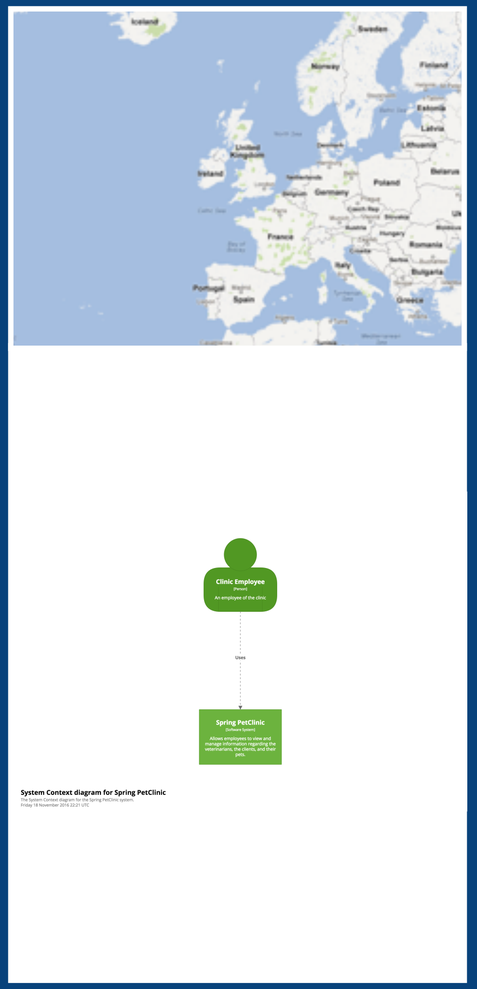
Repository class for Owner domain objects All method names are compliant with Spring Data naming conventions so this interface can easily be exten...

Diagram key

Here are the styles that have been used on this diagram.



Close

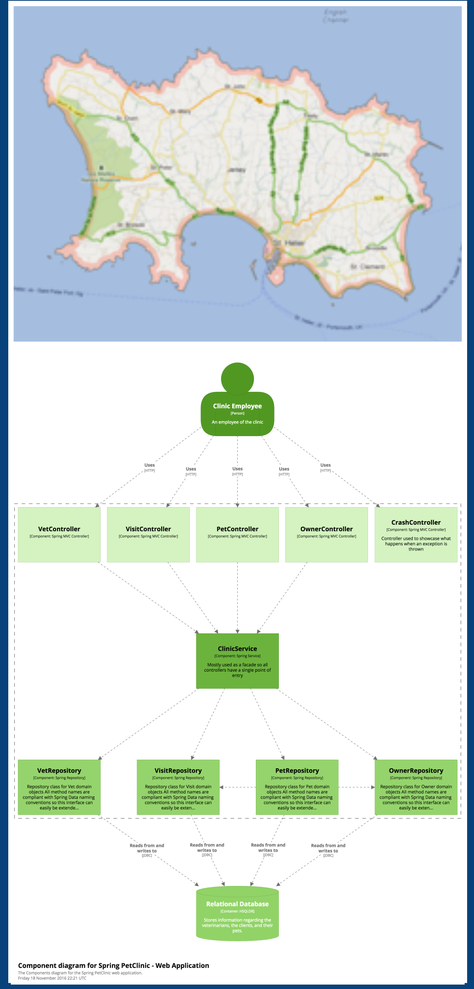


System Context diagram for Spring PetClinic

The System Context diagram for the Spring PetClinic system.
Friday 18 November 2016 22:21 UTC

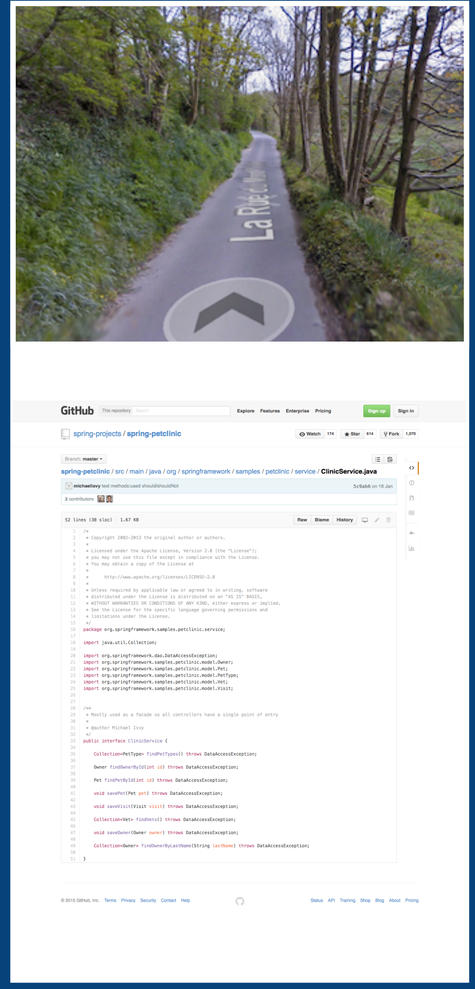
Container diagram for Spring PetClinic

The Containers diagram for the Spring PetClinic system.
Friday 18 November 2016 22:21 UTC



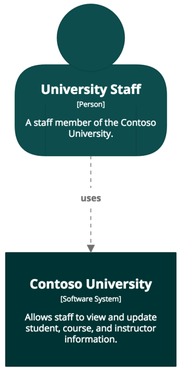
Component diagram for Spring PetClinic - Web Application

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



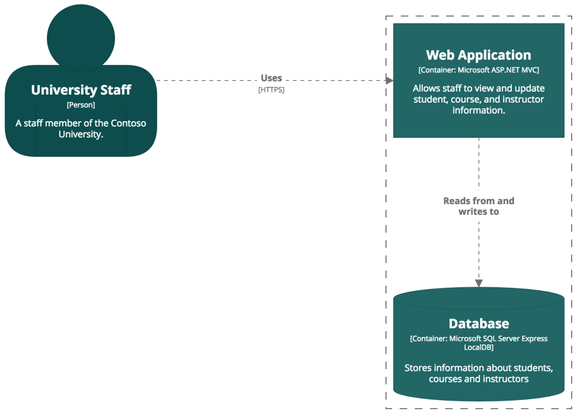
Diagrams are maps

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



System Context diagram for Contoso University

Friday 15 July 2016 11:56 UTC

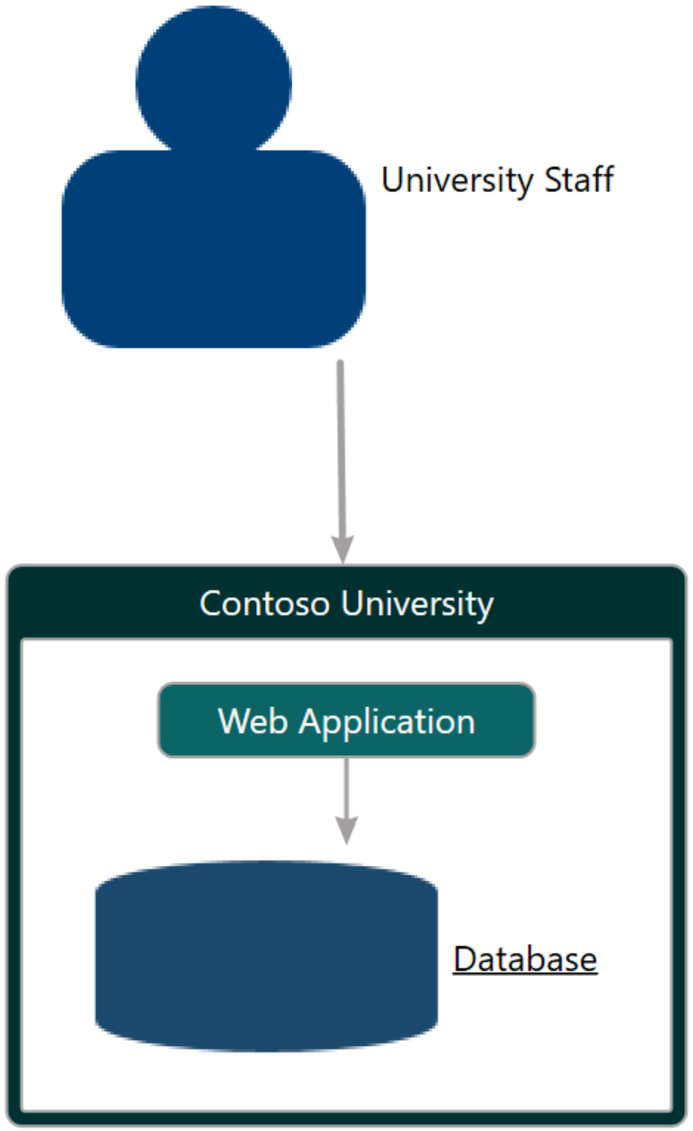


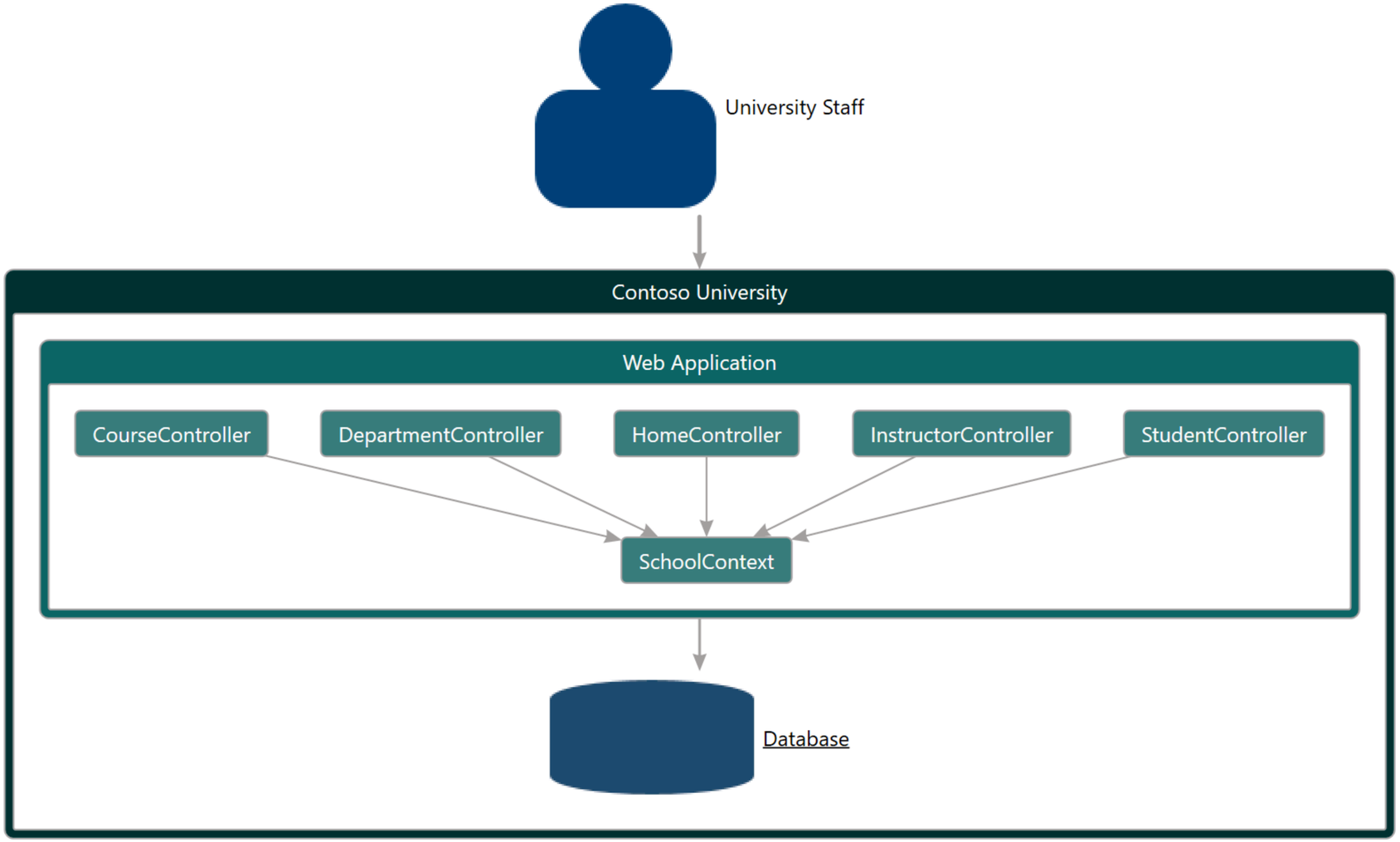
Container diagram for Contoso University

Friday 15 July 2016 11:56 UTC

Component diagram for Contoso University - Web Application

Friday 15 July 2016 11:56 UTC


University Staff
Contoso University
Web Application
Database

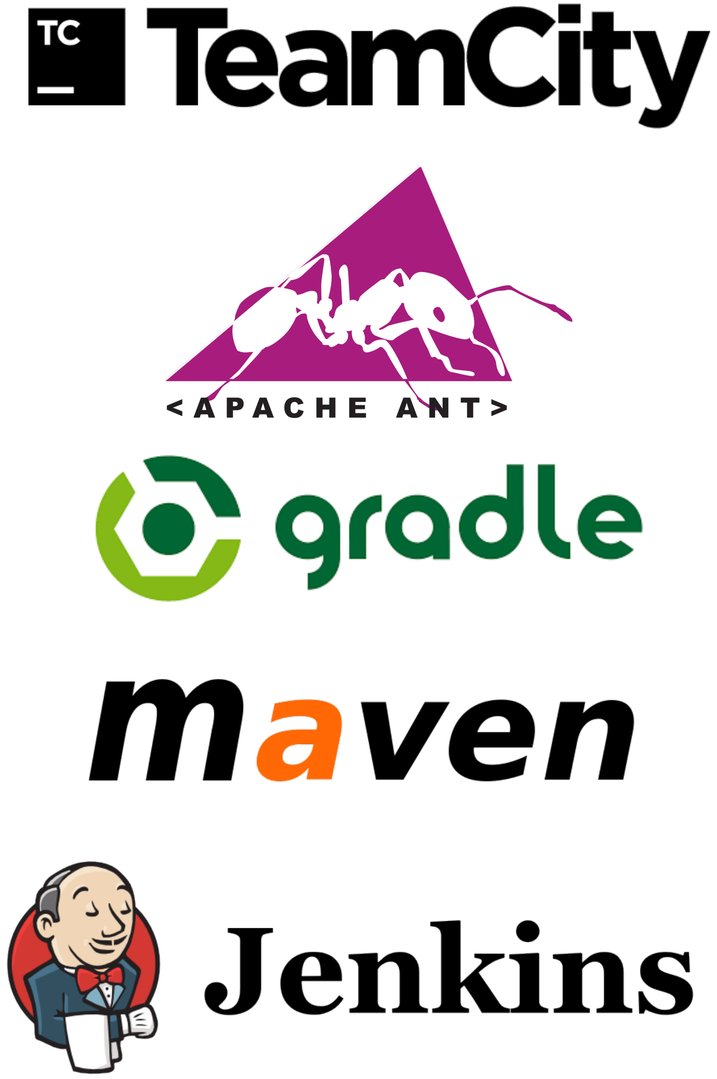

University Staff
Contoso University
Web Application
CourseController
DepartmentController
HomeController
InstructorController
StudentController
SchoolContext
Database

You can also supplement your software architecture model with documentation in Markdown or AsciiDoc

Structurizr API

1. Context

This documentation describes a simple implementation of the Structurizr API, which is designed to be run on-premises to support Structurizr's on-premises API feature. The on-premises API feature provides a way to store workspace data behind your corporate firewall, or using servers anywhere on the Internet, by hosting a local version of the Structurizr API.



Integration with your build process keeps models up to date

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

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

It’s 2017 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