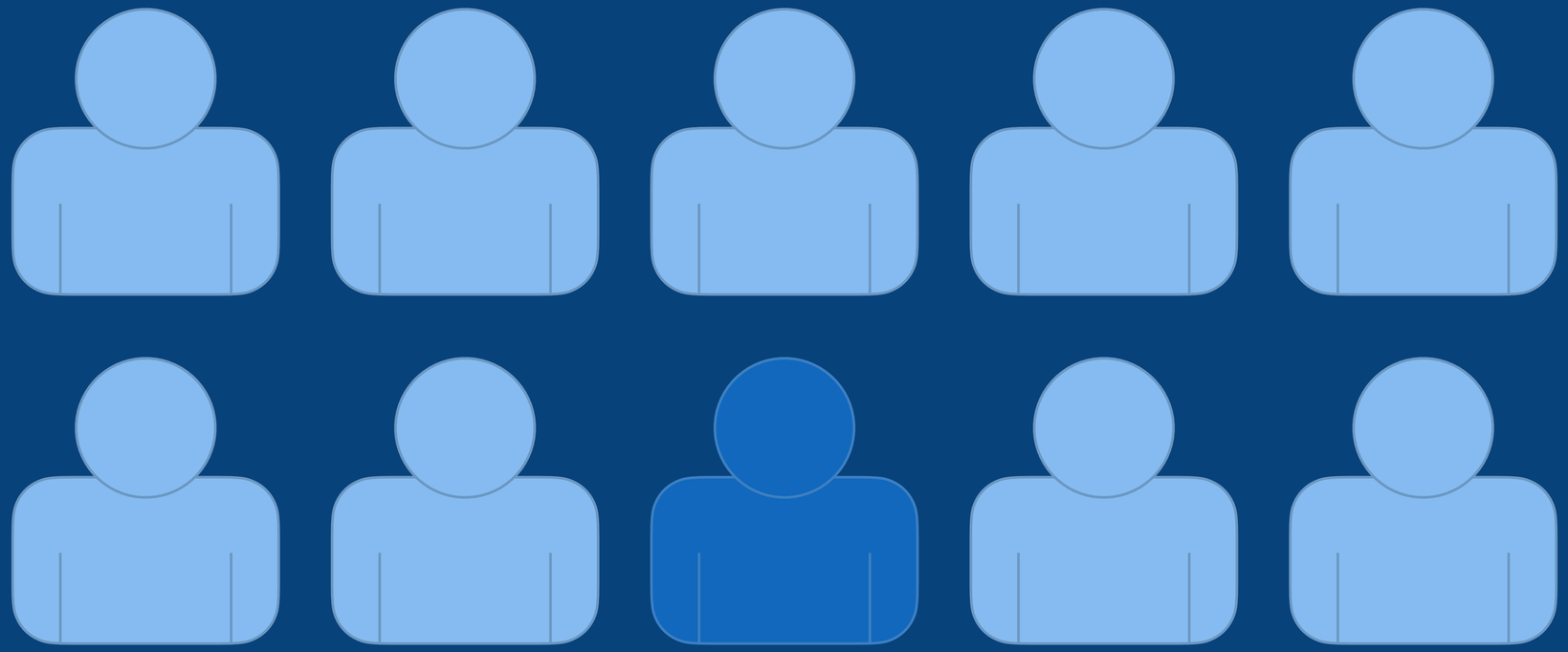
Diagrams as code 2.0

Simon Brown
@simonbrown

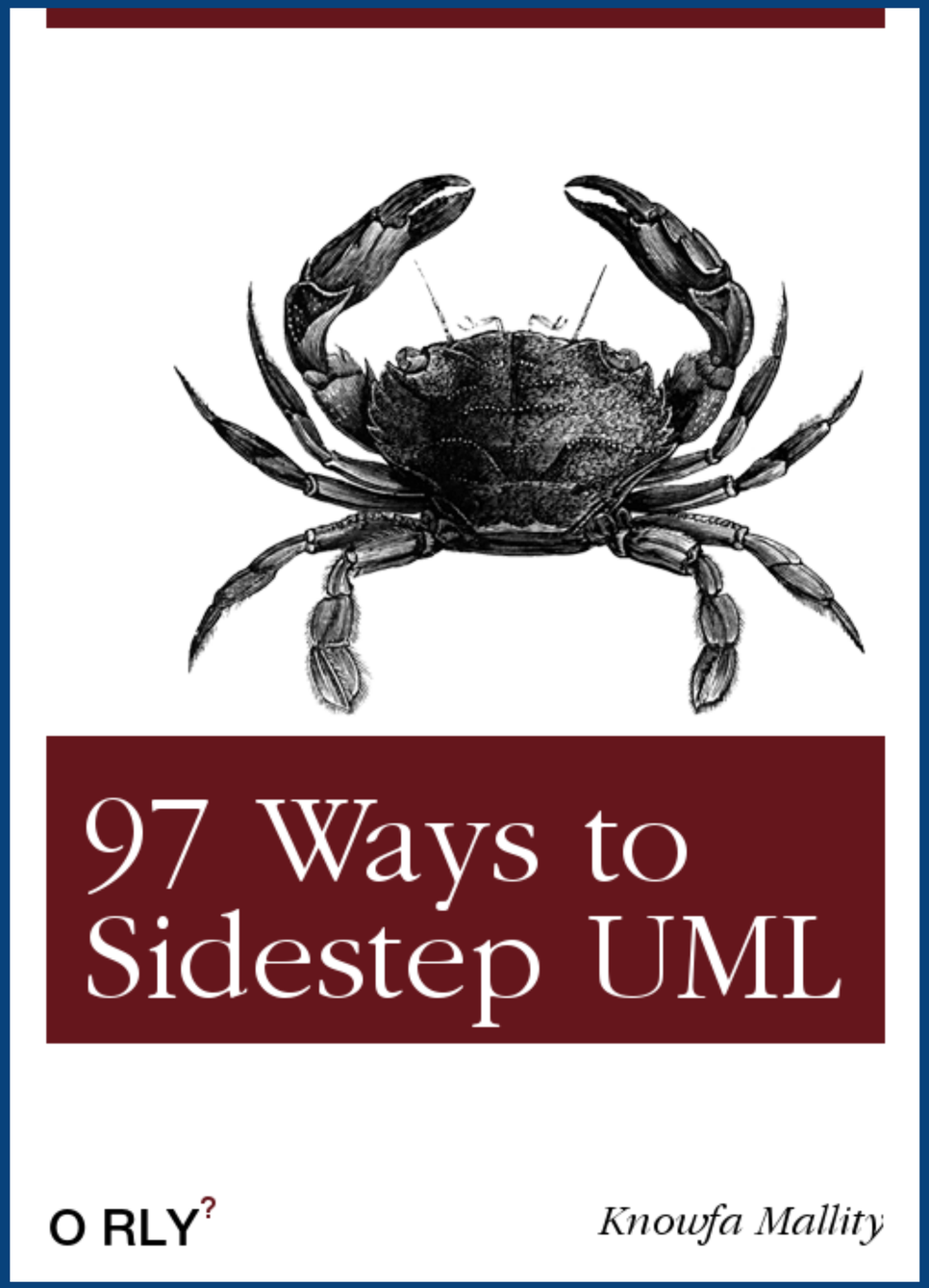
Moving fast in the same direction
as a team requires

good communication

Teams need a **ubiquitous language**
to communicate effectively
(crucial if you’re doing DevOps, DevSecOps, etc)



Fewer people are using UML



#2 “Not everybody else on the team knows it.”

#3 “I’m the only person on the team who knows it.”

#36 “You’ll be seen as old.”

#37 “You’ll be seen as old-fashioned.”

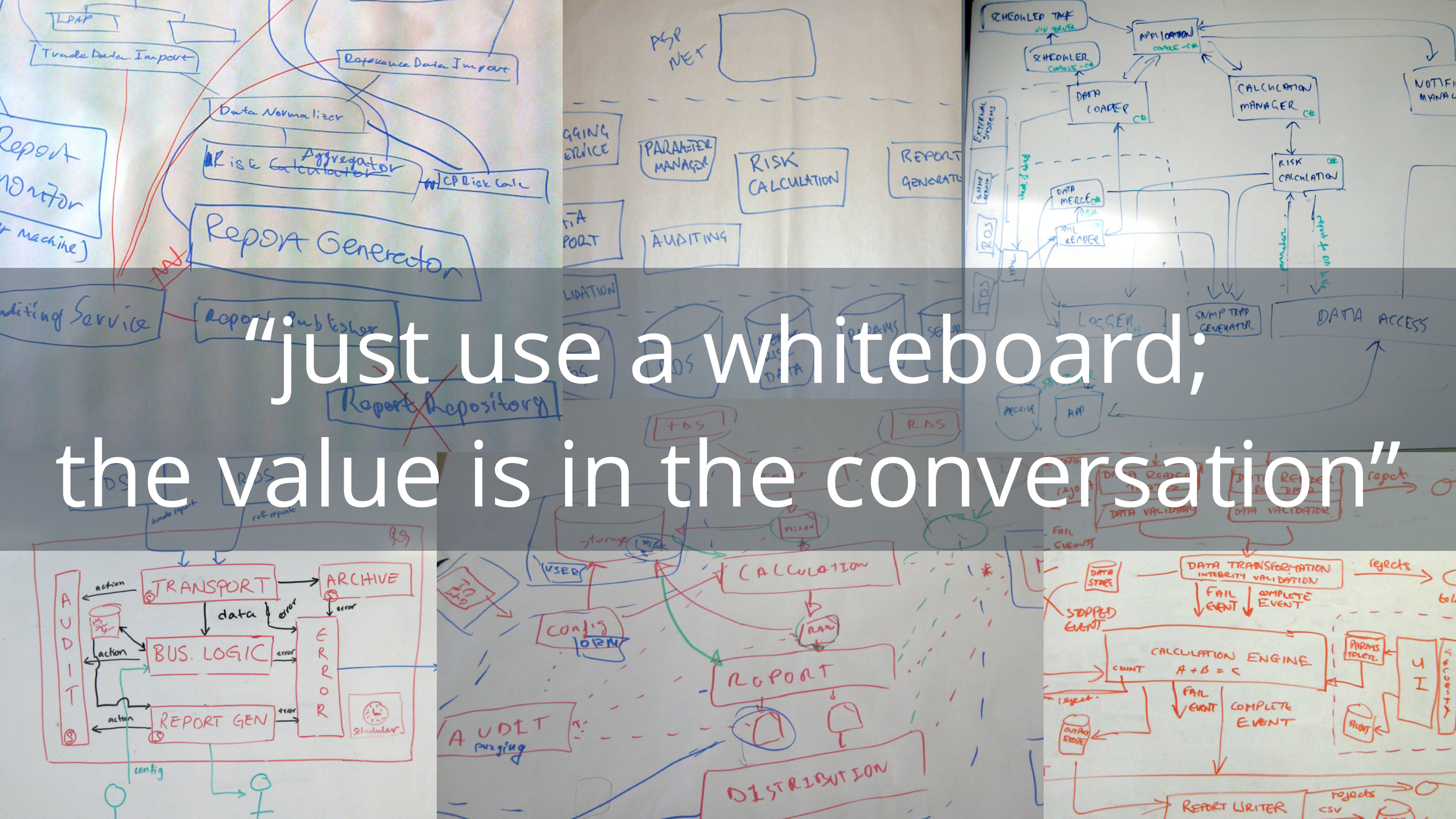
#66 “The tooling sucks.”

#80 “It’s too detailed.”

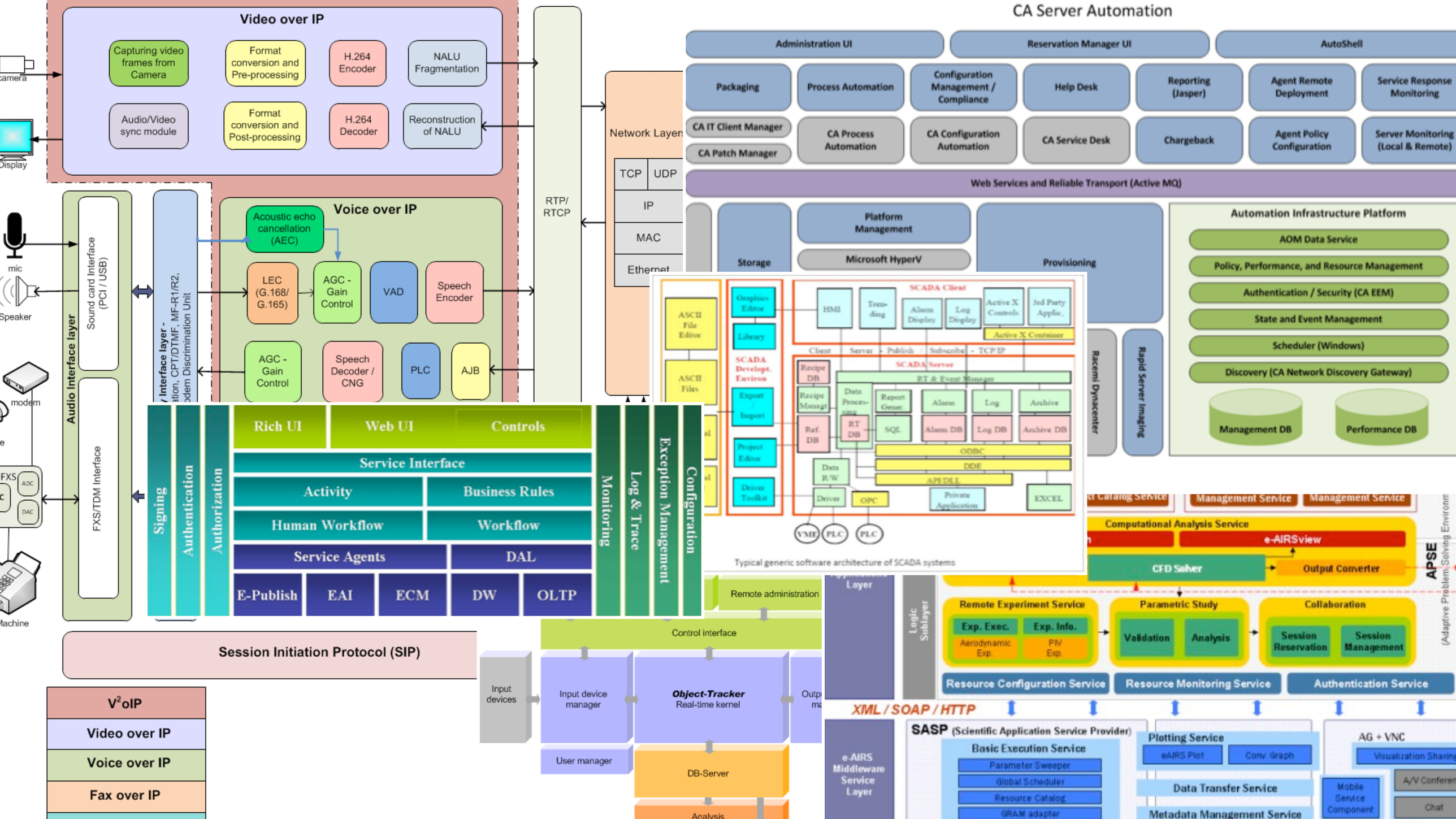
#81 “It’s a very elaborate waste of time.”

#92 “It’s not expected in agile.”

#97 “The value is in the conversation.”



“just use a whiteboard; the value is in the conversation”


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
RTP/ RTCP
Network Layers
TCP
UDP
IP
MAC
Ethernet
Voice over IP
Acoustic echo cancellation (AEC)
LEC (G.168/ G.165)
AGC - Gain Control
VAD
Speech Encoder
AGC - Gain Control
Speech Decoder / CNG
PLC
AJB
mic
Speaker
modem
Audio Interface layer
Sound card Interface (PCI / USB)
FXS/TDM Interface
Machine
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
Automation Infrastructure Platform
AOM Data Service
Policy, Performance, and Resource Management
Authentication / Security (CA EEM)
State and Event Management
Scheduler (Windows)
Discovery (CA Network Discovery Gateway)
Management DB
Performance DB
Racemi Dynacenter
Rapid Server Imaging
ASCII File Editor
ASCII Files
Graphics Editor
Library
SCADA 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 Managt
Ref. DB
Data Process-ing
RT & Event Manager
Report Gener.
Alarm
Log
Archive
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
Logic Sublayer
Computational Analysis Service
e-AIRSview
CFD Solver
Output Converter
APSE
(Adaptive Problem Solving Environment)
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

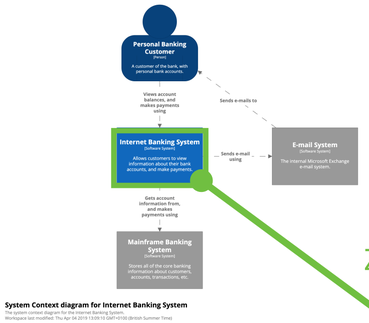
If you’re going to use “boxes & lines”, at least do so in a **structured way**, using a **self-describing notation**

C4

c4model.com

The C4 model for visualising software architecture

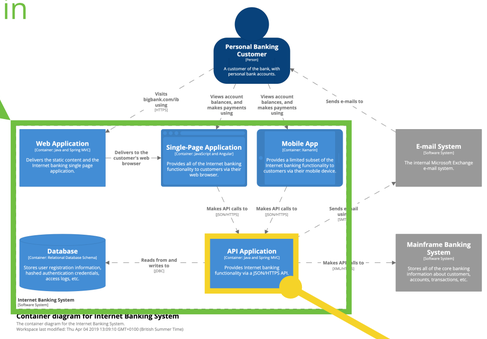
c4model.com



System Context diagram for Internet Banking System

The system context diagram for the Internet Banking System.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)

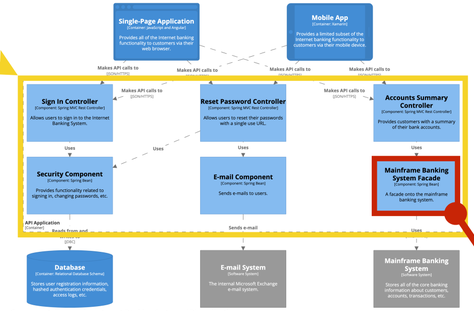
Zoom in



Container diagram for Internet Banking System

The container diagram for the Internet Banking System.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)

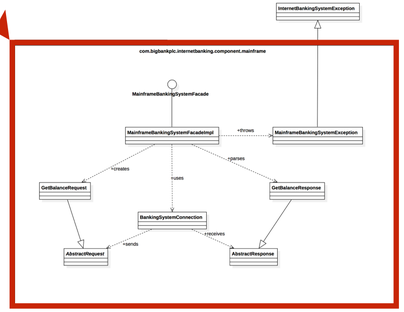
Zoom in



Component diagram for Internet Banking System - API Application

The component diagram for the API Application.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)

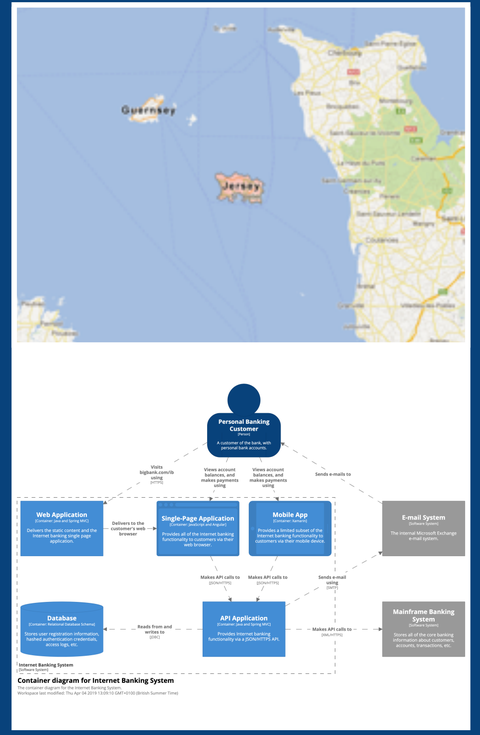
Zoom in



Level 1	Level 2	Level 3	Level 4
Context	Containers	Components	Code

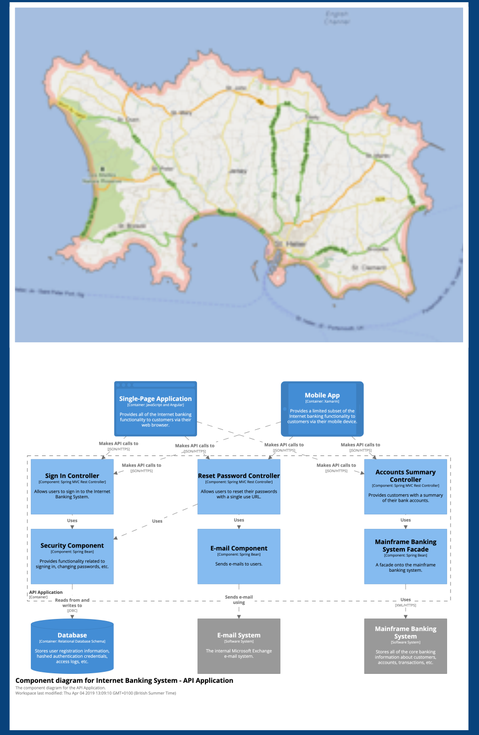
System Context diagram for Internet Banking System

The system context diagram for the Internet Banking System.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)



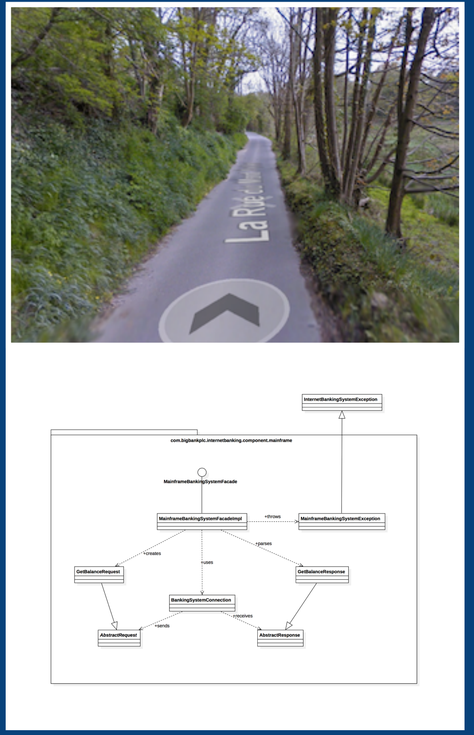
Container diagram for Internet Banking System

The container diagram for the Internet Banking System.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)



Component diagram for Internet Banking System - API Application

The component diagram for the API Application.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)



Diagrams are maps

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

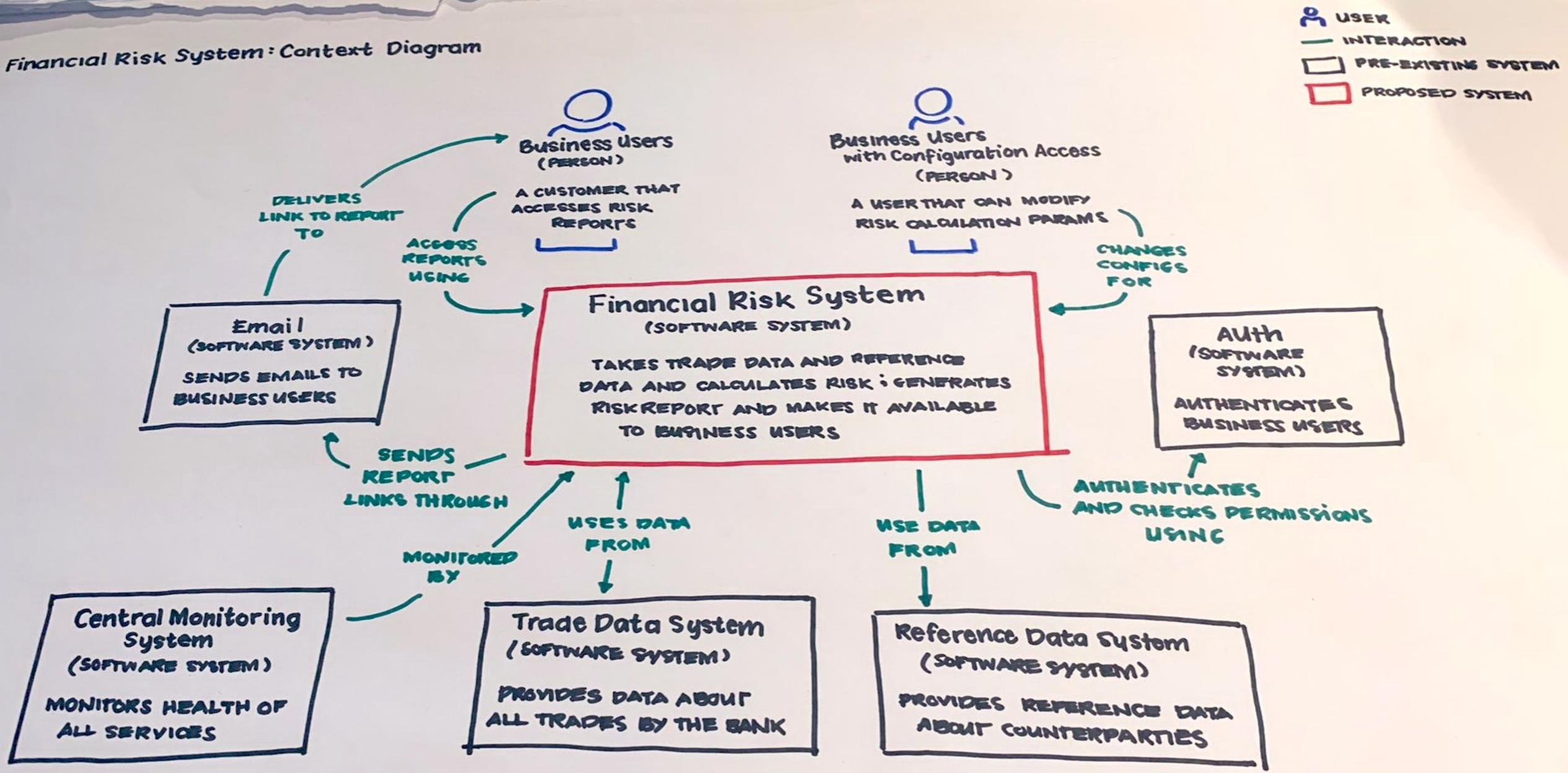
System Context diagram

What is the scope of the software system we're building?

Who is using it? What are they doing?

What system integrations does it need to support?

Financial Risk System: Context Diagram

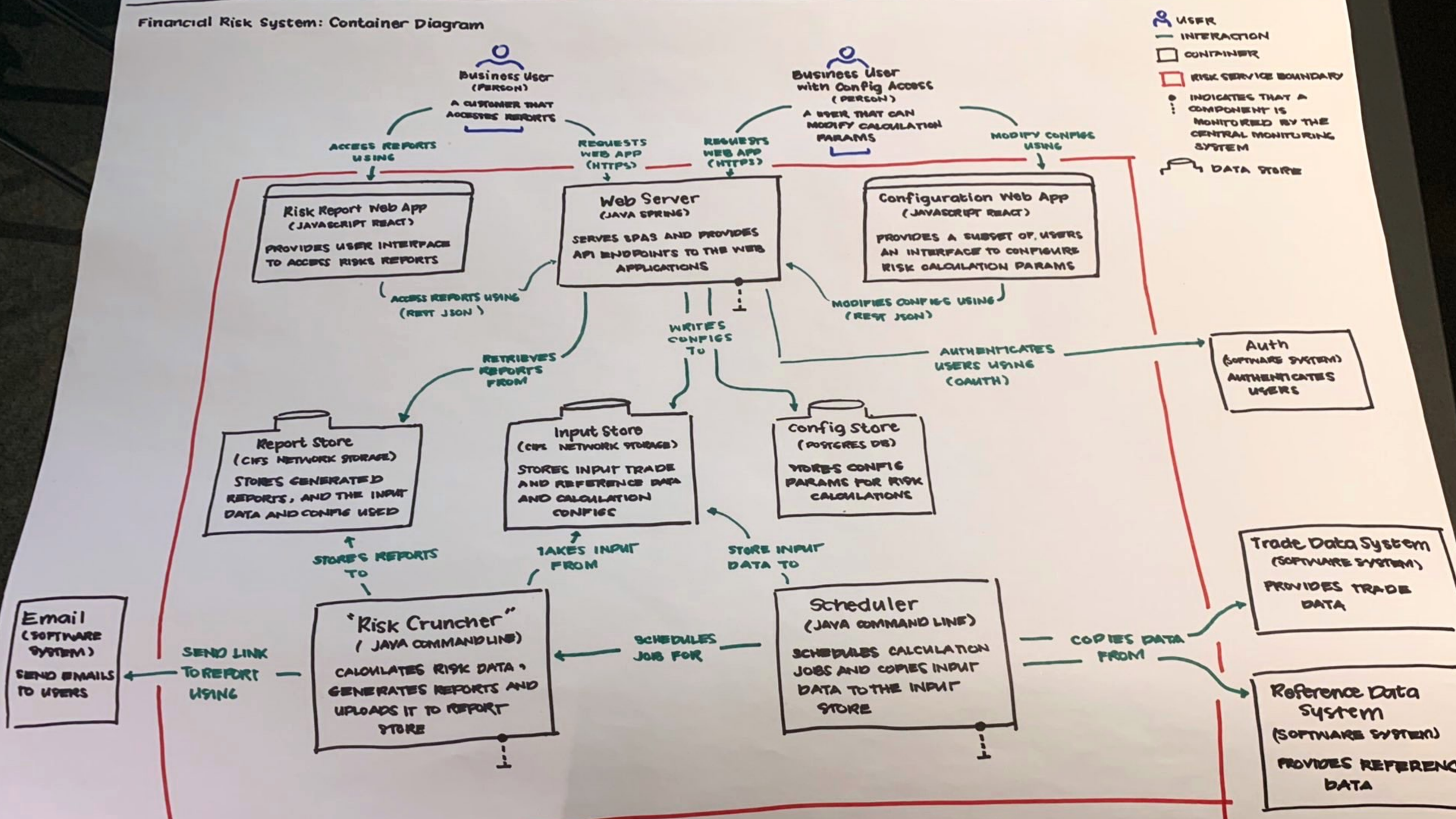


Container diagram

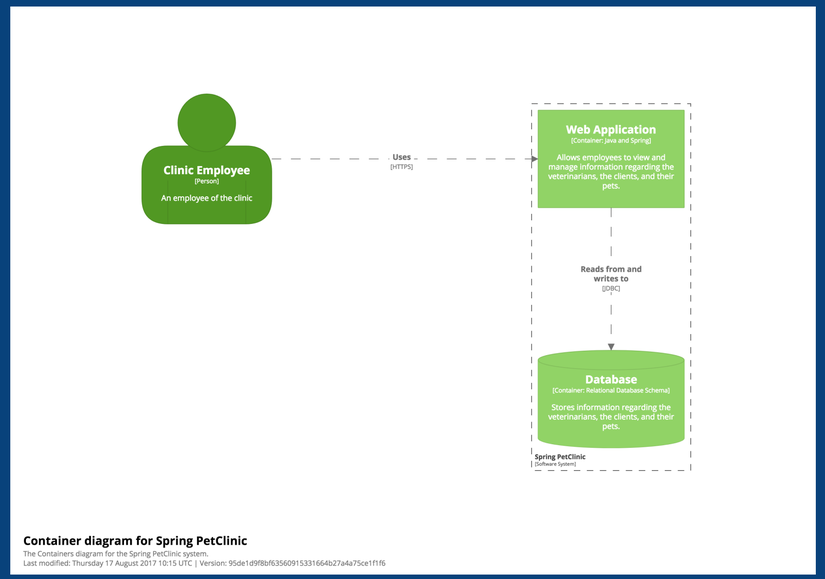
What are the major technology building blocks?

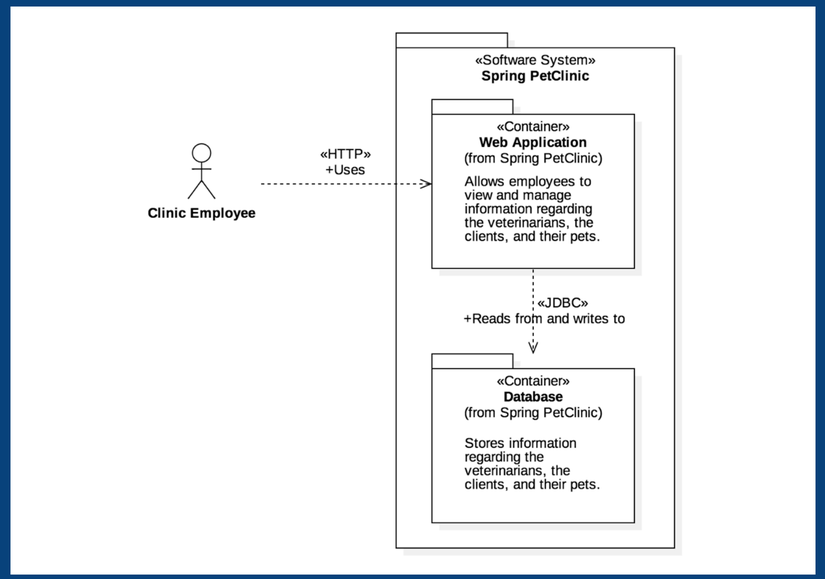
What are their responsibilities?

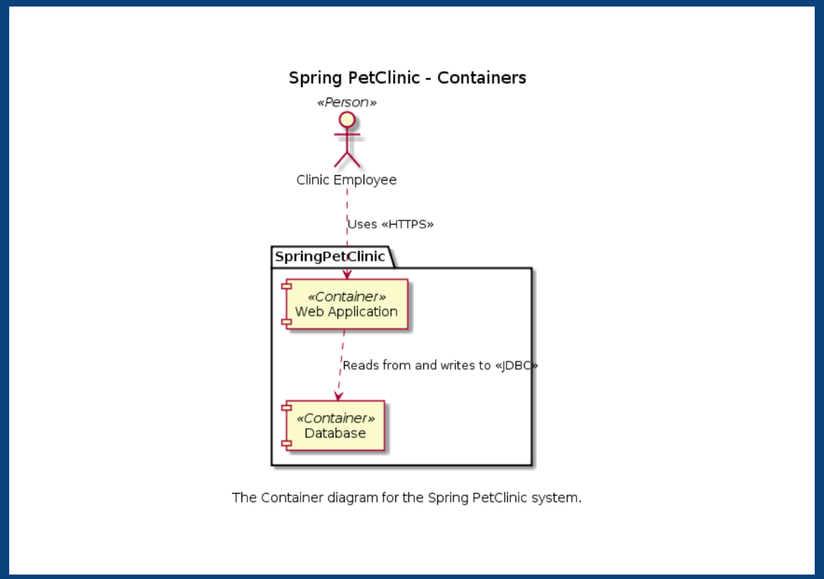
How do they communicate?


Financial Risk System: Container Diagram
USER
INTERACTION
CONTAINER
RISK SERVICE BOUNDARY
INDICATES THAT A COMPONENT IS MONITORED BY THE CENTRAL MONITORING SYSTEM
DATA STORE
Business User
(PERSON)
A CUSTOMER THAT ACCESSES REPORTS
Business User with Config Access
(PERSON)
A USER THAT CAN MODIFY CALCULATION PARAMS
ACCESS REPORTS USING
REQUESTS WEB APP (HTTPS)
REQUESTS WEB APP (HTTPS)
MODIFY CONFIGS USING
Risk Report Web App
(JAVASCRIPT REACT)
PROVIDES USER INTERFACE TO ACCESS RISK REPORTS
Web Server
(JAVA SPRING)
SERVES SPAS AND PROVIDES API ENDPOINTS TO THE WEB APPLICATIONS
Configuration Web App
(JAVASCRIPT REACT)
PROVIDES A SUBSET OF USERS AN INTERFACE TO CONFIGURE RISK CALCULATION PARAMS
ACCESS REPORTS USING (REST JSON)
MODIFIES CONFIGS USING (REST JSON)
WRITES CONFIGS TO
RETRIEVES REPORTS FROM
AUTHENTICATES USERS USING (OAUTH)
Auth
(SOFTWARE SYSTEM)
AUTHENTICATES USERS
Report Store
(CIFS NETWORK STORAGE)
STORES GENERATED REPORTS, AND THE INPUT DATA AND CONFIG USED
Input Store
(CIFS NETWORK STORAGE)
STORES INPUT TRADE AND REFERENCE DATA AND CALCULATION CONFIGS
Config Store
(POSTGRES DB)
STORES CONFIG PARAMS FOR RISK CALCULATIONS
STORES REPORTS TO
TAKES INPUT FROM
STORE INPUT DATA TO
Email
(SOFTWARE SYSTEM)
SEND EMAILS TO USERS
SEND LINK TO REPORT USING
"Risk Cruncher"
(JAVA COMMANDLINE)
CALCULATES RISK DATA, GENERATES REPORTS AND UPLOADS IT TO REPORT STORE
SCHEDULES JOB FOR
Scheduler
(JAVA COMMAND LINE)
SCHEDULES CALCULATION JOBS AND COPIES INPUT DATA TO THE INPUT STORE
COPIES DATA FROM
Trade Data System
(SOFTWARE SYSTEM)
PROVIDES TRADE DATA
Reference Data System
(SOFTWARE SYSTEM)
PROVIDES REFERENC
DATA

The C4 model is **notation independent**







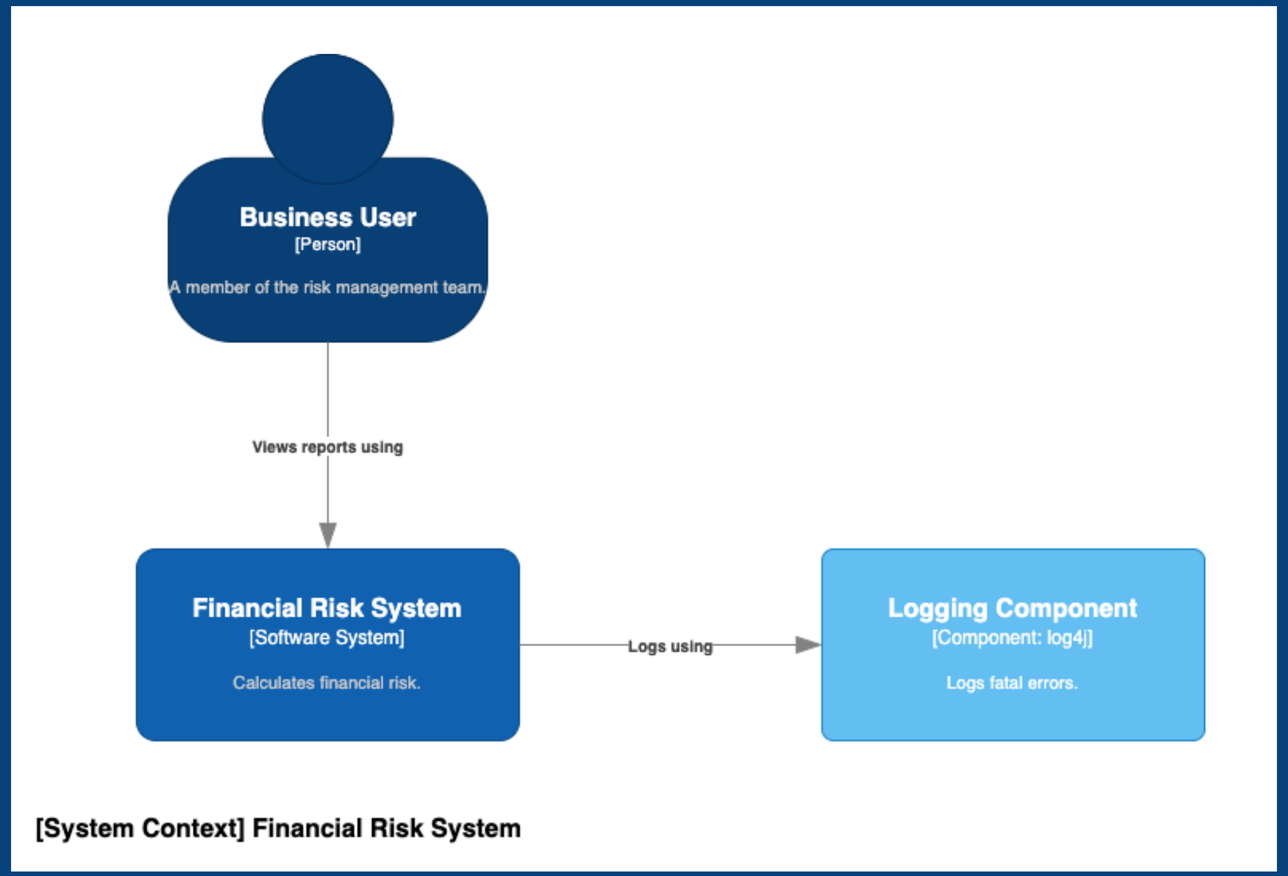
A **common set of abstractions**
is more important
than a common notation

Tooling?

Most teams use general purpose diagramming tools

(Visio, diagrams.net, Lucidchart, Gliffy, etc)

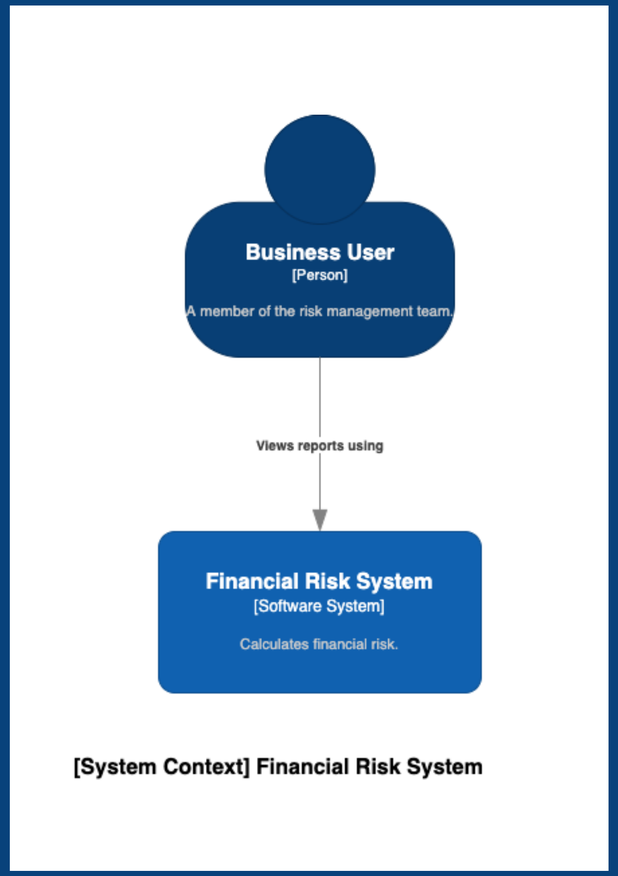
Visio, diagrams.net, Lucidchart, Gliffy, etc
- **not recommended** for
software architecture diagrams

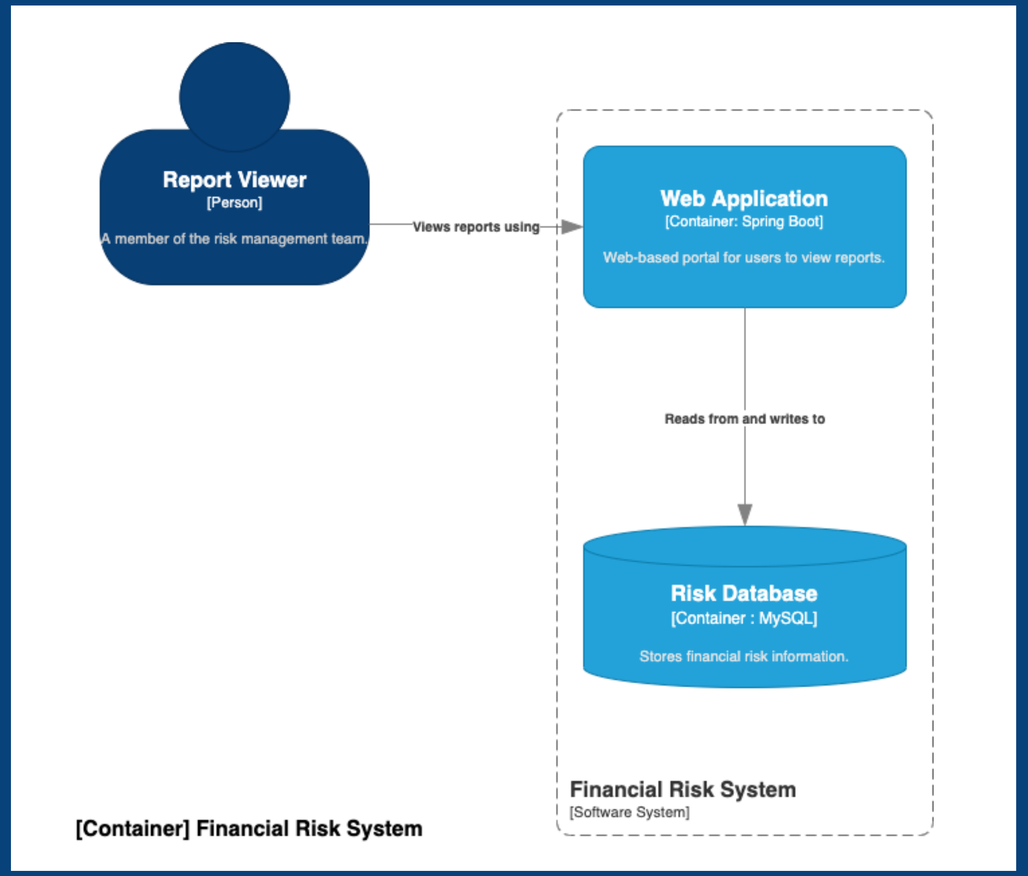


No guidance, rules, semantics, etc

<mxfile host="app.diagrams.net" modified="2022-05-16T09:48:26.450Z" agent="5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safari/537.36" etag="3crCYGp9Lf1JWI7VV2hT" version="16.5.2" type="device"><diagram id="ZBMMDNz2AIJuiLfg3vVe" name="Page-1">7Zjbcts2EIafRpfW8CDS8qV1cHKRdjJV0taXEAmRiEGCBSFL6tN3FwRI8CDF6XQ6mUl8GHF/
YJe74OIz4Vm4Ls7vJKnyX0RK+Szw0vMs3MyCwPeXMXygcmmUey9qhEyy1EzqhB37mxrRM+qRpbTuTVRCcMWqvpiIsqSJ6mlESnHqTzsI3r9rRTI6EnYJ4WP1D5aqvFGXwX2nv6csy+2d/
fihGSmInWwqqXOSipMjhdtZuJZCqOaqOK8px8Wz69L4PV0ZbROTtFQTDmL/BdcDZnCS0BzqpnKwZItfSWHK/
FxTadVPl8qoH8FFlFbf0DqRrFIMJD38CPJRO3rigCVfsExxUCdICi8vtaLFvPHnZA+tof1mQcwh5dVBlJhgrS5mteO/jsIO3NW6F/
Aeflydu0G4ysynjrLvhMiUFER2EJZjP3QArblzX07ZaytFTTC9EhAs2ji+7jybgrwV7Fur9W9Va6IkggvZizILwkR/TflG/
afXW5+ptRiWGTiN97h98opVGLI63HzwPtPq+ff3d37b0G2jdoUGuSrw0fsmf7PLfbRTUucUI3tgnHKmaF1Bv4JwAqCgA+N83VYbesvw6T4CvVZSvFB3JA79aG1u4egH/
YUeOdGNXZwzZNU8Wcwr3eCQ+YpwlmFbJ7CboKPDVUEV2aZMmbwrwUql90+E7eHNYQnXnv7F7liD5qNl9L523xeN4A/
EKS2YECdD9u8NP+FKUmgqstcPwWsf4yuVip6vMsR9kO+ogFWQuKuNw9KwzMD8wZinjoyBZ7TcpaL1I4ZAWRu5AxZcmN6xpkHYv8DZTpMHxFLjYMi1XccoO/
MG4lJ35zSk+wm5HxFywU3ISXEsU40y36Js10eZg0HdKCuSvGTazeKqFCUdM8/3Yn/l3SDbCF5EJhay3iQrN5H34I8pN0TGiHrBGHtTKGz5NGZhMAXI7wKajhb0SfofkHMxQGdru+xcTLEz+F/
Y2cHxN8oJbqA6Z9VVLsL7Yn0VeXpXTTAAVk/dmV5FCHB6UNch8NYopunfiM6rhJgi6LfhIbyJB1qmj3geAWvPRfLyKWdlnwnuq9HEbl94+N3uZnsaQU+I/
cS4jTPa7csAv3Ee3+sEsJMZnHHGAIBINoWFDkWkcoUvx6LamaIAMlYxM2JESItBzJWmcJoy84VUuchESfi2U1f0zNSfOHseGevZ5gLXm7MNhMbFMeBswiB5xF2jlbAnnEBoPrtjXShtXVxrGKx52pj87e2ObwJ
HmdCvvRnDoZXIjKo3/nFx8DGFB6l36Gs/tyk8mHAfkeIditrXNoOiMB4QpqnJeLlH0EGgKPxKoKboUSBNq7bGNwAMzO6c3Hh3/20It/8A</diagram></mxfile>

Content and presentation is mixed





No model, no consistency

<mxfile host="app.diagrams.net" modified="2022-05-16T09:48:26.450Z" agent="5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safari/537.36" etag="3crCYGp9Lf1JWI7VV2hT" version="16.5.2" type="device"><diagram id="ZBMMDNz2AIJuiLfg3vVe" name="Page-1">7Zjbcts2EIafRpfW8CDS8qV1cHKRdjJV0taXEAmRiEGCBSFL6tN3FwRI8CDF6XQ6mUl8GHF/
YJe74OIz4Vm4Ls7vJKnyX0RK+Szw0vMs3MyCwPeXMXygcmmUey9qhEyy1EzqhB37mxrRM+qRpbTuTVRCcMWqvpiIsqSJ6mlESnHqTzsI3r9rRTI6EnYJ4WP1D5aqvFGXwX2nv6csy+2d/
fihGSmInWwqqXOSipMjhdtZuJZCqOaqOK8px8Wz69L4PV0ZbROTtFQTDmL/BdcDZnCS0BzqpnKwZItfSWHK/
FxTadVPl8qoH8FFlFbf0DqRrFIMJD38CPJRO3rigCVfsExxUCdICi8vtaLFvPHnZA+tof1mQcwh5dVBlJhgrS5mteO/jsIO3NW6F/
Aeflydu0G4ysynjrLvhMiUFER2EJZjP3QArblzX07ZaytFTTC9EhAs2ji+7jybgrwV7Fur9W9Va6IkggvZizILwkR/TflG/
afXW5+ptRiWGTiN97h98opVGLI63HzwPtPq+ff3d37b0G2jdoUGuSrw0fsmf7PLfbRTUucUI3tgnHKmaF1Bv4JwAqCgA+N83VYbesvw6T4CvVZSvFB3JA79aG1u4egH/
YUeOdGNXZwzZNU8Wcwr3eCQ+YpwlmFbJ7CboKPDVUEV2aZMmbwrwUql90+E7eHNYQnXnv7F7liD5qNl9L523xeN4A/
EKS2YECdD9u8NP+FKUmgqstcPwWsf4yuVip6vMsR9kO+ogFWQuKuNw9KwzMD8wZinjoyBZ7TcpaL1I4ZAWRu5AxZcmN6xpkHYv8DZTpMHxFLjYMi1XccoO/
MG4lJ35zSk+wm5HxFywU3ISXEsU40y36Js10eZg0HdKCuSvGTazeKqFCUdM8/3Yn/l3SDbCF5EJhay3iQrN5H34I8pN0TGiHrBGHtTKGz5NGZhMAXI7wKajhb0SfofkHMxQGdru+xcTLEz+F/
Y2cHxN8oJbqA6Z9VVLsL7Yn0VeXpXTTAAVk/dmV5FCHB6UNch8NYopunfiM6rhJgi6LfhIbyJB1qmj3geAWvPRfLyKWdlnwnuq9HEbl94+N3uZnsaQU+I/
cS4jTPa7csAv3Ee3+sEsJMZnHHGAIBINoWFDkWkcoUvx6LamaIAMlYxM2JESItBzJWmcJoy84VUuchESfi2U1f0zNSfOHseGevZ5gLXm7MNhMbFMeBswiB5xF2jlbAnnEBoPrtjXShtXVxrGKx52pj87e2ObwJ
HmdCvvRnDoZXIjKo3/nFx8DGFB6l36Gs/tyk8mHAfkeIditrXNoOiMB4QpqnJeLlH0EGgKPxKoKboUSBNq7bGNwAMzO6c3Hh3/20It/8A</diagram></mxfile>

Hard to diff

Limited opportunities for automation

Time consuming

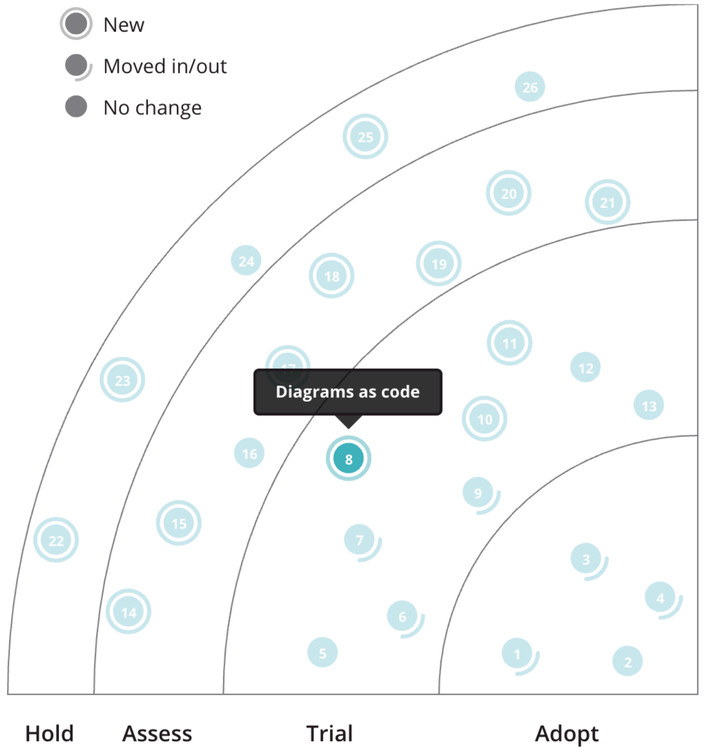
TECHNOLOGY RADAR

Techniques

Trial

5. Continuous delivery for machine learning (CD4ML)
6. Data mesh
7. Declarative data pipeline definition
8. Diagrams as code

We're seeing more and more tools that enable you to create software architecture and other **diagrams as code**. There are benefits to using these tools over the heavier alternatives, including easy version control and the ability to generate the DSLs from many sources. Tools in this space that we like include **Diagrams**, **Structurizr DSL**, **AsciiDoctor Diagram** and stables such as **WebSequenceDiagrams**, **PlantUML** and the venerable **Graphviz**. It's also fairly simple to generate your own SVG these days, so don't rule out quickly writing your own tool either.



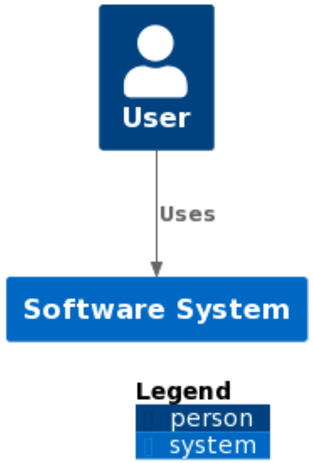
Unable to find something you expected to see?

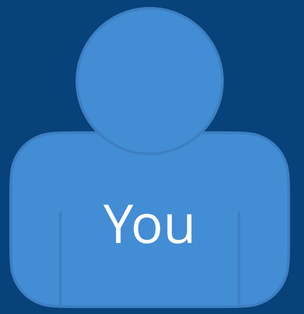
Each edition of the radar features blips reflecting what we came across during the previous six months. We might have covered

“Diagrams as code” is easy to author, diff, version control, collaborate on, integrate into CI/CD, etc

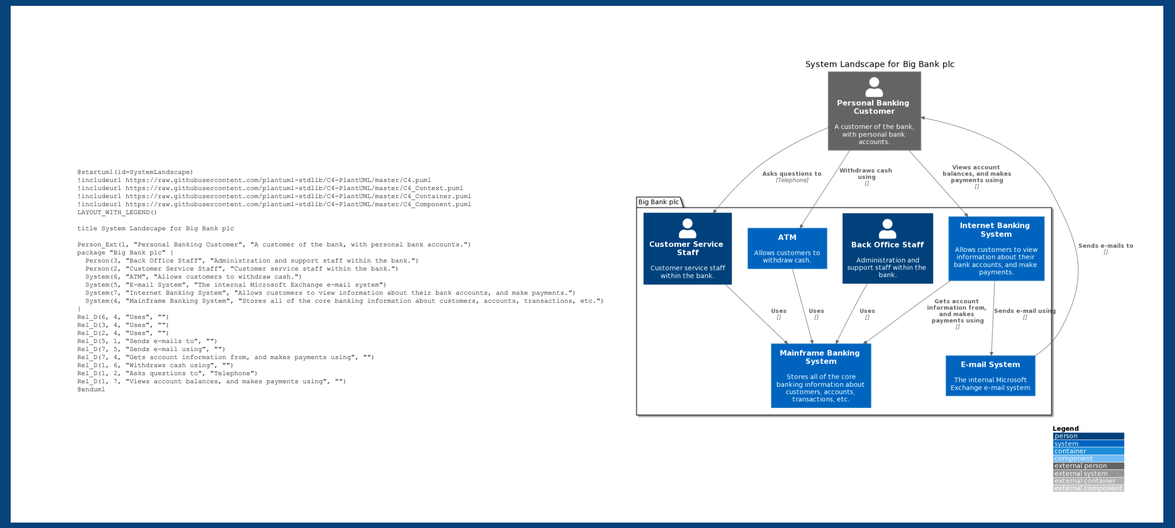
C4-PlantUML

```
@startuml
!include https://raw.githubusercontent.com/plantuml-stdlib/C4-PlantUML/master/C4.puml
!include https://raw.githubusercontent.com/plantuml-stdlib/C4-PlantUML/master/C4_Context.puml

Person(User, "User")
System(SoftwareSystem, "Software System")

Rel_D(User, SoftwareSystem, "Uses")

SHOW_LEGEND()
@enduml
```

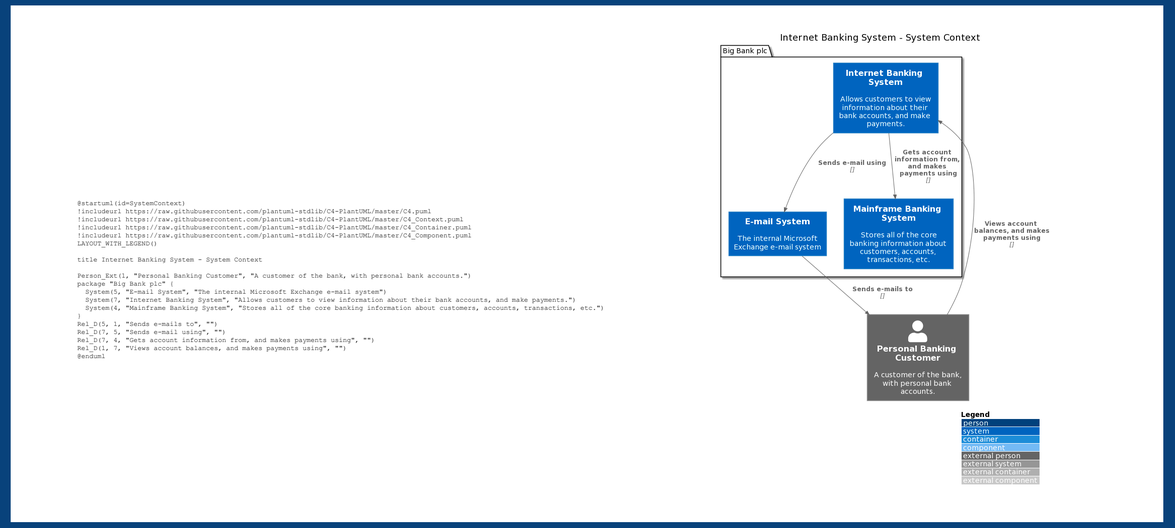




Create and maintain



```
@startuml(id=SystemLandscape)
!includeurl https://raw.githubusercontent.com/plantuml-stdlib/C4-PlantUML/master/C4.puml
!includeurl https://raw.githubusercontent.com/plantuml-stdlib/C4-PlantUML/master/C4_Context.puml
!includeurl https://raw.githubusercontent.com/plantuml-stdlib/C4-PlantUML/master/C4_Container.puml
!includeurl https://raw.githubusercontent.com/plantuml-stdlib/C4-PlantUML/master/C4_Component.puml
LAYOUT_WITH_LEGEND()

title System Landscape for Big Bank plc

Person_Ext(1, "Personal Banking Customer", "A customer of the bank, with personal bank accounts.")
package "Big Bank plc" {
  Person(3, "Back Office Staff", "Administration and support staff within the bank.")
  Person(2, "Customer Service Staff", "Customer service staff within the bank.")
  System(6, "ATM", "Allows customers to withdraw cash.")
  System(5, "E-mail System", "The internal Microsoft Exchange e-mail system")
  System(7, "Internet Banking System", "Allows customers to view information about their bank accounts, and make payments.")
  System(4, "Mainframe Banking System", "Stores all of the core banking information about customers, accounts, transactions, etc.")
}
Rel_D(6, 4, "Uses", "")
Rel_D(3, 4, "Uses", "")
Rel_D(2, 4, "Uses", "")
Rel_D(5, 1, "Sends e-mails to", "")
Rel_D(7, 5, "Sends e-mail using", "")
Rel_D(7, 4, "Gets account information from, and makes payments using", "")
Rel_D(1, 6, "Withdraws cash using", "")
Rel_D(1, 2, "Asks questions to", "Telephone")
Rel_D(1, 7, "Views account balances, and makes payments using", "")
@enduml
```

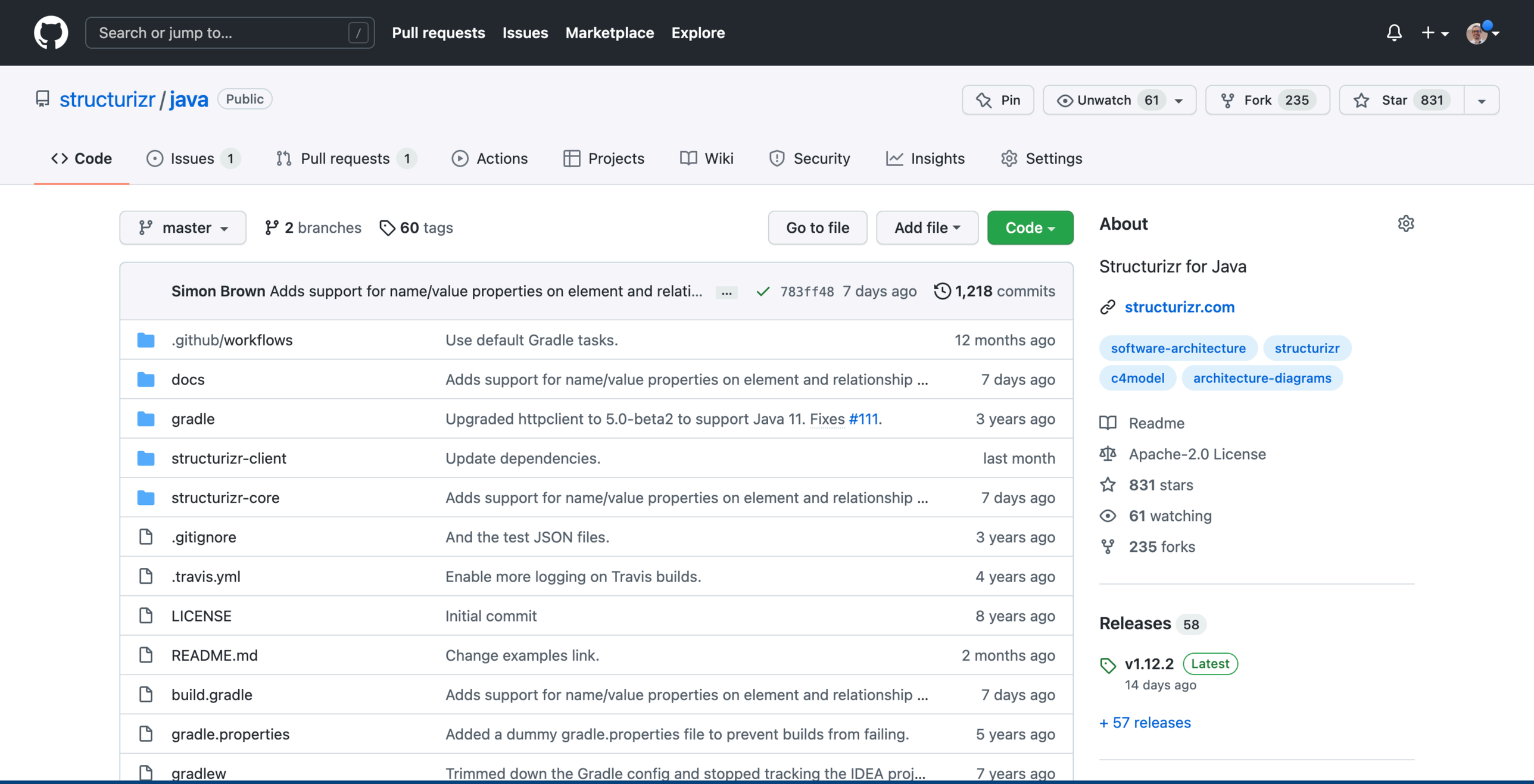



```
@startuml(id=SystemContext)
!includeurl https://raw.githubusercontent.com/plantuml-stdlib/C4-PlantUML/master/C4.puml
!includeurl https://raw.githubusercontent.com/plantuml-stdlib/C4-PlantUML/master/C4_Context.puml
!includeurl https://raw.githubusercontent.com/plantuml-stdlib/C4-PlantUML/master/C4_Container.puml
!includeurl https://raw.githubusercontent.com/plantuml-stdlib/C4-PlantUML/master/C4_Component.puml
LAYOUT_WITH_LEGEND()

title Internet Banking System - System Context

Person_Ext(1, "Personal Banking Customer", "A customer of the bank, with personal bank accounts.")
package "Big Bank plc" {
  System(5, "E-mail System", "The internal Microsoft Exchange e-mail system")
  System(7, "Internet Banking System", "Allows customers to view information about their bank accounts, and make payments.")
  System(4, "Mainframe Banking System", "Stores all of the core banking information about customers, accounts, transactions, etc.")
}
Rel_D(5, 1, "Sends e-mails to", "")
Rel_D(7, 5, "Sends e-mail using", "")
Rel_D(7, 4, "Gets account information from, and makes payments using", "")
Rel_D(1, 7, "Views account balances, and makes payments using", "")
@enduml
```


Diagrams as code 1.0

You create and maintain multiple diagrams, remembering to keep them all in sync whenever you change a diagram

From diagramming to modelling



https://github.com/structurizr/java

Workspace

A workspace is the wrapper for a software architecture model, views, and supplementary documentation.

Model

The definition of the software architecture model - elements and the relationships between them.

Views

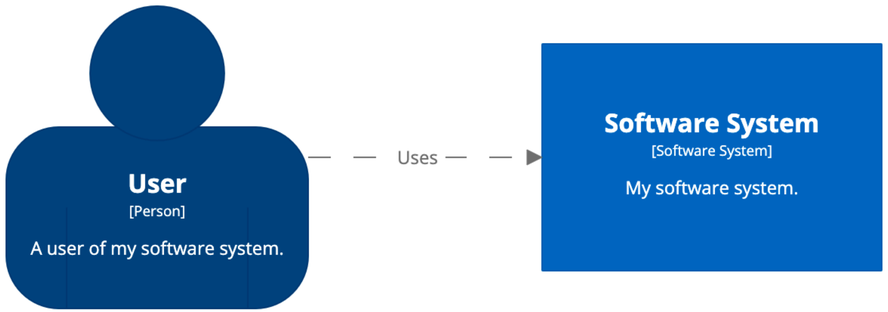
The definition of views of the software architecture model.

Documentation

Supplementary documentation (Markdown/AsciiDoc content and accompanying images) plus architecture decision records (ADRs).

Structurizr for Java

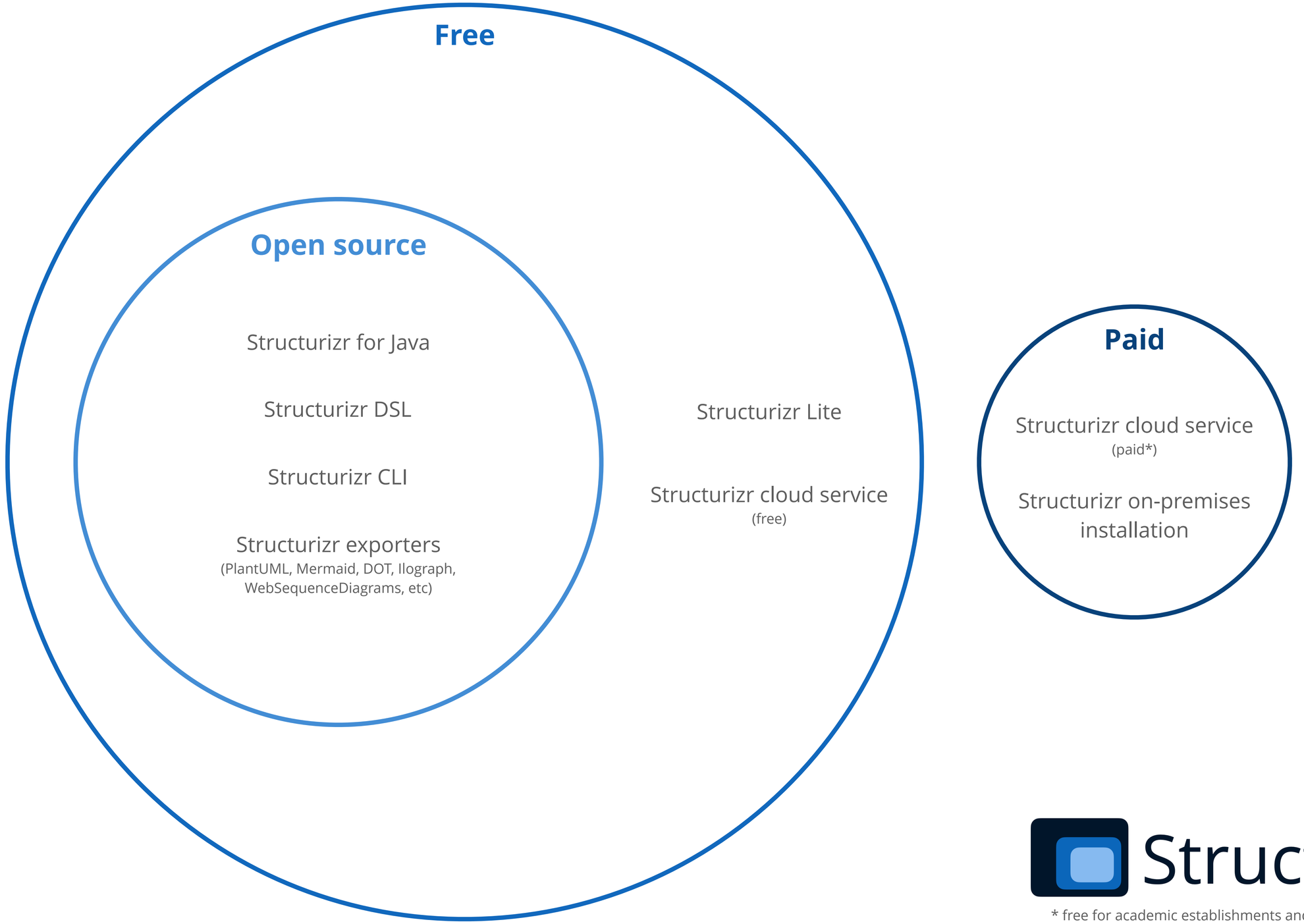
```
public static void main(String[] args) throws Exception {

    Workspace workspace = new Workspace("Getting Started", "This is a model of my software system.");

    Model model = workspace.getModel();
    Person user = model.addPerson("User", "A user of my software system.");
    SoftwareSystem softwareSystem = model.addSoftwareSystem("Software System", "My software system.");
    user.uses(softwareSystem, "Uses");

    ViewSet views = workspace.getViews();
    SystemContextView contextView = views.createSystemContextView(softwareSystem, "SystemContext", "...");
    contextView.addAllSoftwareSystems();
    contextView.addAllPeople();

    Styles styles = views.getConfiguration().getStyles();
    styles.addElementStyle(Tags.SOFTWARE_SYSTEM).background("#1168bd").color("#ffffff");
    styles.addElementStyle(Tags.PERSON).background("#08427b").color("#ffffff").shape(Shape.Person);

    ...
}
```



+ some free and open source tooling for creating software architecture diagrams

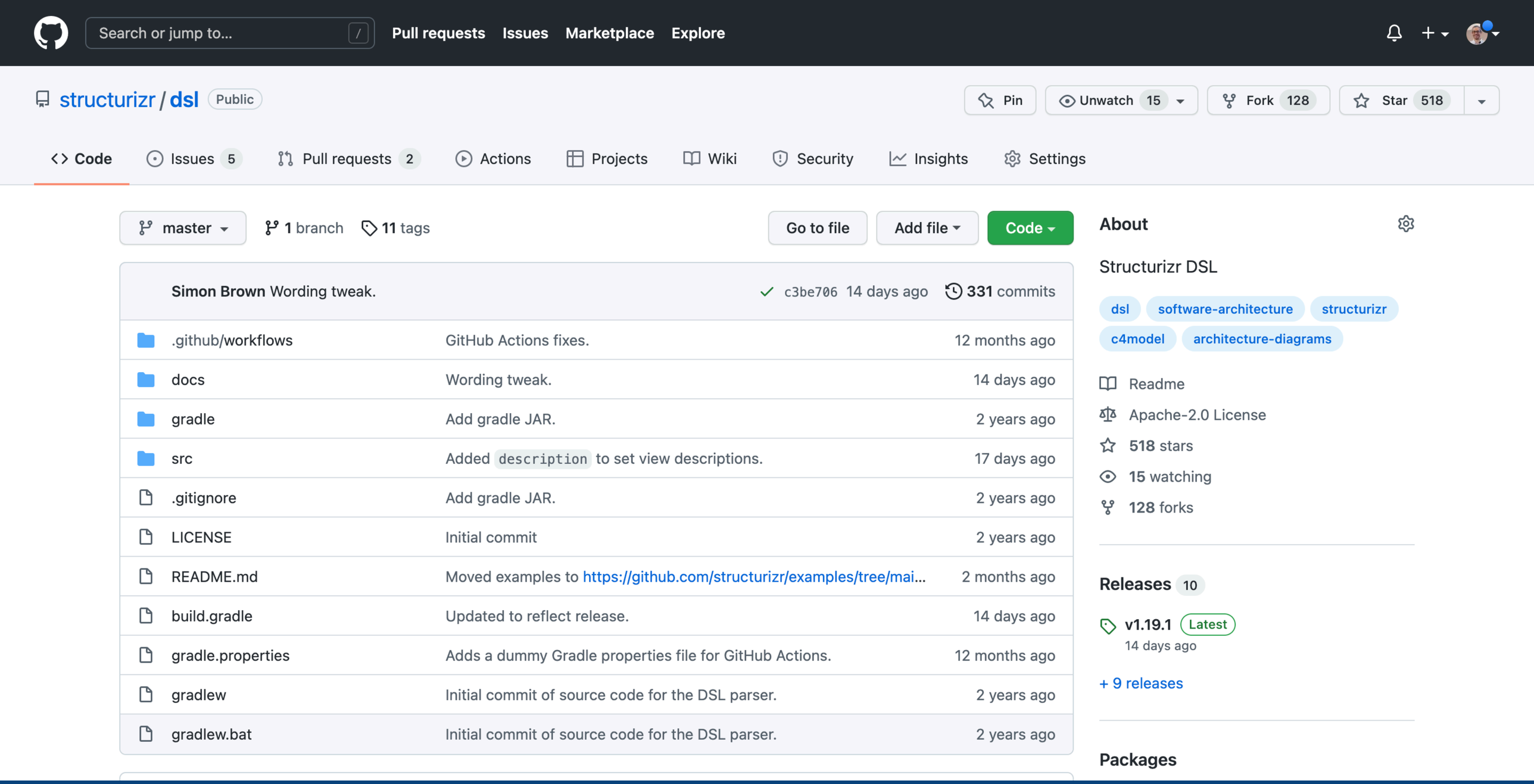
Structurizr DSL

A text-based domain specific language (DSL) to create software architecture diagrams based upon the C4 model

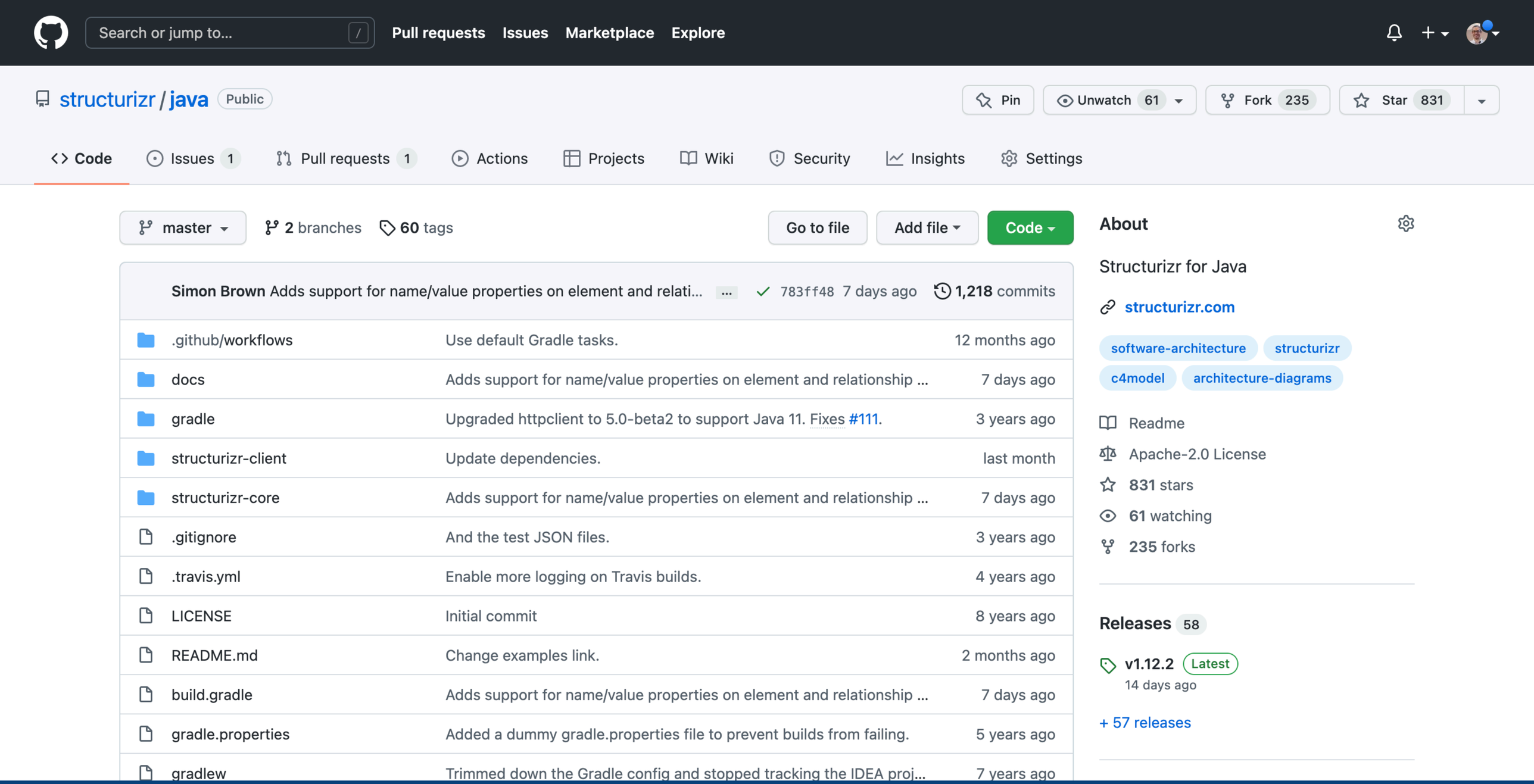


Structurizr

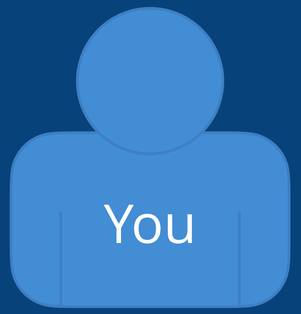
* free for academic establishments and open source projects



https://github.com/structurizr/dsl



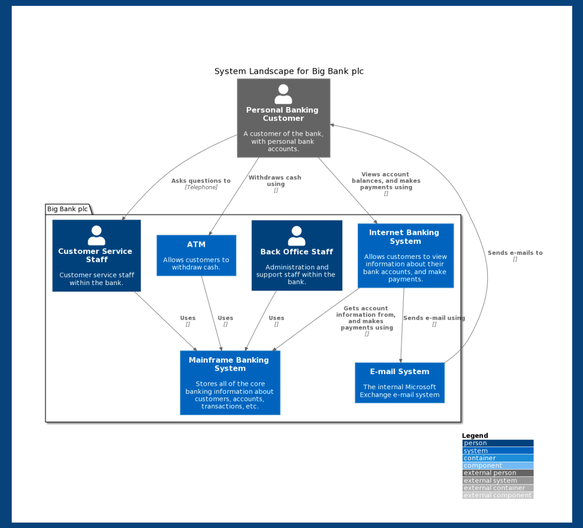
https://github.com/structurizr/java

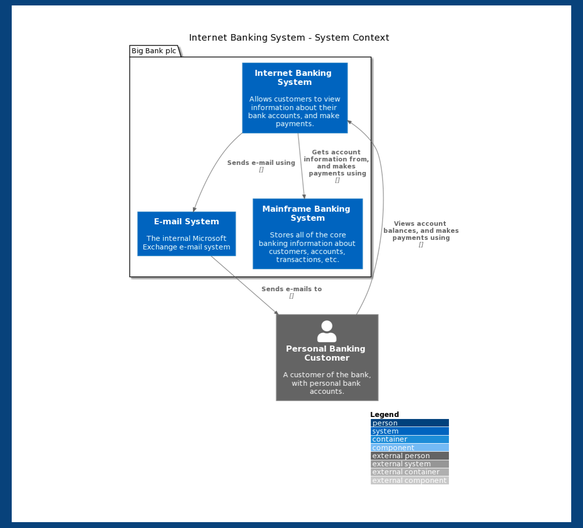


Create and maintain

```
workspace "Big Bank plc" "This is an example workspace to illustrate the key features of Structurizr, via the DSL, based around a fictional online banking system." {

    model {
        customer = person "Personal Banking Customer" "A customer of the bank, with personal bank accounts."

        enterprise "Big Bank plc" {
            supportStaff = person "Customer Service Staff" "Customer service staff within the bank." "Bank Staff"
            backoffice = person "Back Office Staff" "Administration and support staff within the bank." "Bank Staff"

            mainframe = softwaresystem "Mainframe Banking System" "Stores all of the core banking information about customers, accounts, transactions, etc." "Existing System"
            email = softwaresystem "E-mail System" "The internal Microsoft Exchange e-mail system" "Existing System"
            atm = softwaresystem "ATM" "Allows customers to withdraw cash." "Existing System"

            internetBankingSystem = softwaresystem "Internet Banking System" "Allows customers to view information about their bank accounts, and make payments." {
                singlePageApplication = container "Single-Page Application" "Provides all of the Internet banking functionality to customers via their web browser." "JavaScript and Angular" "Web Browser"
                mobileApp = container "Mobile App" "Provides a limited subset of the Internet banking functionality to customers via their mobile device." "Xamarin" "Mobile App"
                webApplication = container "Web Application" "Delivers the static content and the Internet banking single page application." "Java and Spring MVC"
                apiApplication = container "API Application" "Provides Internet banking functionality via a JSON/HTTPS API." "Java and Spring MVC" {
                    signinController = component "Sign In Controller" "Allows users to sign in to the Internet Banking System." "Spring MVC Rest Controller"
                    accountsSummaryController = component "Accounts Summary Controller" "Provides customers with a summary of their bank accounts." "Spring MVC Rest Controller"
                    resetPasswordController = component "Reset Password Controller" "Allows users to reset their passwords with a single use URL." "Spring MVC Rest Controller"
                    securityComponent = component "Security Component" "Provides functionality related to signing in, changing passwords, etc." "Spring Bean"
                    mainframeBankingSystemFacade = component "Mainframe Banking System Facade" "A facade onto the mainframe banking system." "Spring Bean"
                    emailComponent = component "E-mail Component" "Sends e-mails to users." "Spring Bean"
                }
                database = container "Database" "Stores user registration information, hashed authentication credentials, access logs, etc." "Oracle Database Schema" "Database"
            }

        }

        # relationships between people and software systems
        user -> customer -> internetBankingSystem "Views account balances, and makes payments using"
        internetBankingSystem -> mainframe "Gets account information from, and makes payments using"
        internetBankingSystem -> email "Sends e-mail using"
        email -> customer "Sends e-mails to"
        customer -> supportStaff "Asks questions to" "Telephone"
        supportStaff -> mainframe "Uses"
        customer -> atm "Withdraws cash using"
        atm -> mainframe "Uses"
        backoffice -> mainframe "Uses"

        # relationships to/from containers
        customer -> webApplication "Visits bigbank.com/ib using" "HTTPS"
        customer -> singlePageApplication "Views account balances, and makes payments using"
        customer -> mobileApp "Views account balances, and makes payments using"
        webApplication -> singlePageApplication "Delivers to the customer's web browser"

        # relationships to/from components
        singlePageApplication -> signinController "Makes API calls to" "JSON/HTTPS"
        singlePageApplication -> accountsSummaryController "Makes API calls to" "JSON/HTTPS"
        singlePageApplication -> resetPasswordController "Makes API calls to" "JSON/HTTPS"
        mobileApp -> signinController "Makes API calls to" "JSON/HTTPS"
        mobileApp -> accountsSummaryController "Makes API calls to" "JSON/HTTPS"
        mobileApp -> resetPasswordController "Makes API calls to" "JSON/HTTPS"
        signinController -> securityComponent "Uses"
        accountsSummaryController -> mainframeBankingSystemFacade "Uses"
        resetPasswordController -> securityComponent "Uses"
        resetPasswordController -> emailComponent "Uses"
        securityComponent -> database "Reads from and writes to" "JDBC"
        mainframeBankingSystemFacade -> mainframe "Makes API calls to" "XML/HTTPS"
        emailComponent -> email "Sends e-mails using"
    }

}
```

Automatically generates





Diagrams as code 2.0

You create and maintain a single model, and the tool generates multiple diagrams, automatically keeping them all in sync whenever you change the model

“Models as code”?

Domain concepts

(not “boxes and lines”)

```
@startuml
title Software System - System Context

top to bottom direction

hide stereotype

rectangle "==User\n<size:10>[Person]</size>" <<User>> as User
rectangle "==Software System\n<size:10>[Software System]</size>" <<SoftwareSystem>> as SoftwareSystem

User ..> SoftwareSystem : "Uses"
@enduml
```

Domain language of diagramming

(no rules, no guidance)

```
@startuml
!include https://raw.githubusercontent.com/plantuml-stdlib/C4-PlantUML/master/C4.puml
!include https://raw.githubusercontent.com/plantuml-stdlib/C4-PlantUML/master/C4_Context.puml

Person(User, "User")
System(SoftwareSystem, "Software System")

Rel_D(User, SoftwareSystem, "Uses")

SHOW_LEGEND()
@enduml
```

Domain language of software architecture

(still no rules, no guidance)

```
workspace {

    model {
        user = person "User"
        softwareSystem = softwareSystem "Software System"

        user -> softwareSystem "Uses"
    }

    views {
        systemContext softwareSystem {
            include *
            autoLayout
        }
    }

}
```

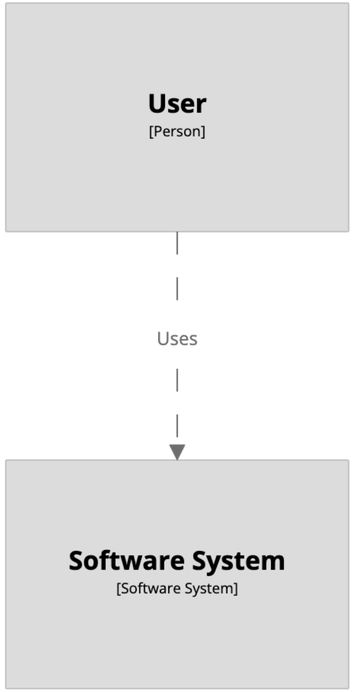
Domain language of software architecture

(metamodel and rules)

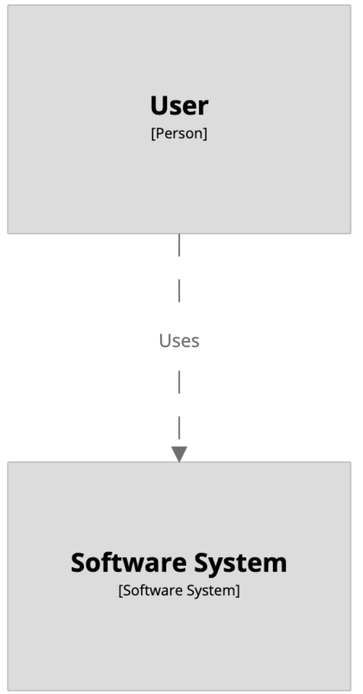
Model-based

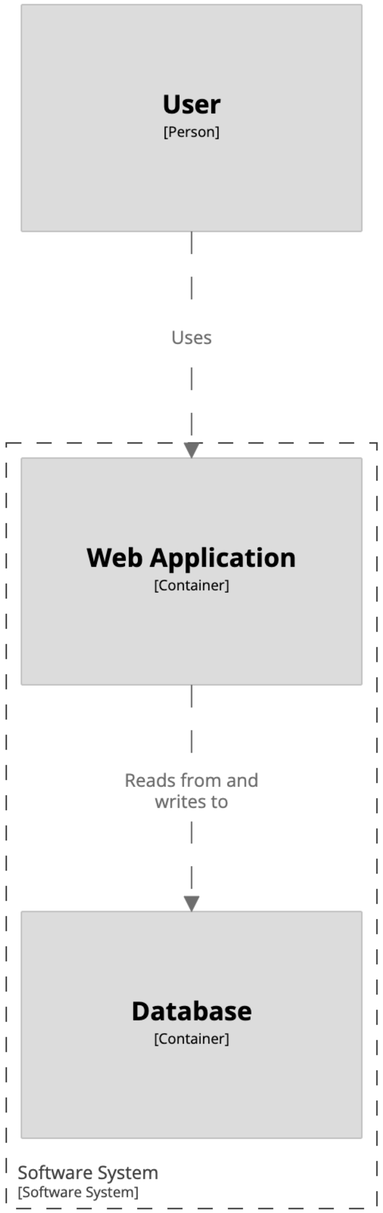
(DRY)

```
workspace {

    model {
        user = person "User"
        softwareSystem = softwareSystem "Software System"



        user -> softwareSystem "Uses"

    }

    views {
        systemContext softwareSystem {
            include *
            autoLayout
        }



    }

}
```



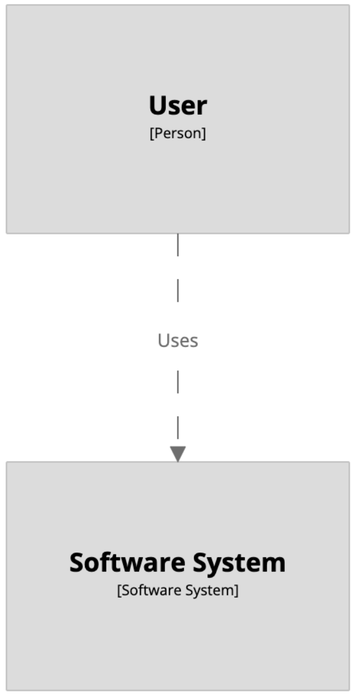
```
workspace {

    model {
        user = person "User"
        softwareSystem = softwareSystem "Software System" {
            webapp = container "Web Application"
            database = container "Database"
        }

        user -> webapp "Uses"
        webapp -> database "Reads from and writes to"
    }

    views {
        systemContext softwareSystem {
            include *
            autoLayout
        }

        container softwareSystem {
            include *
            autolayout
        }
    }

}
```



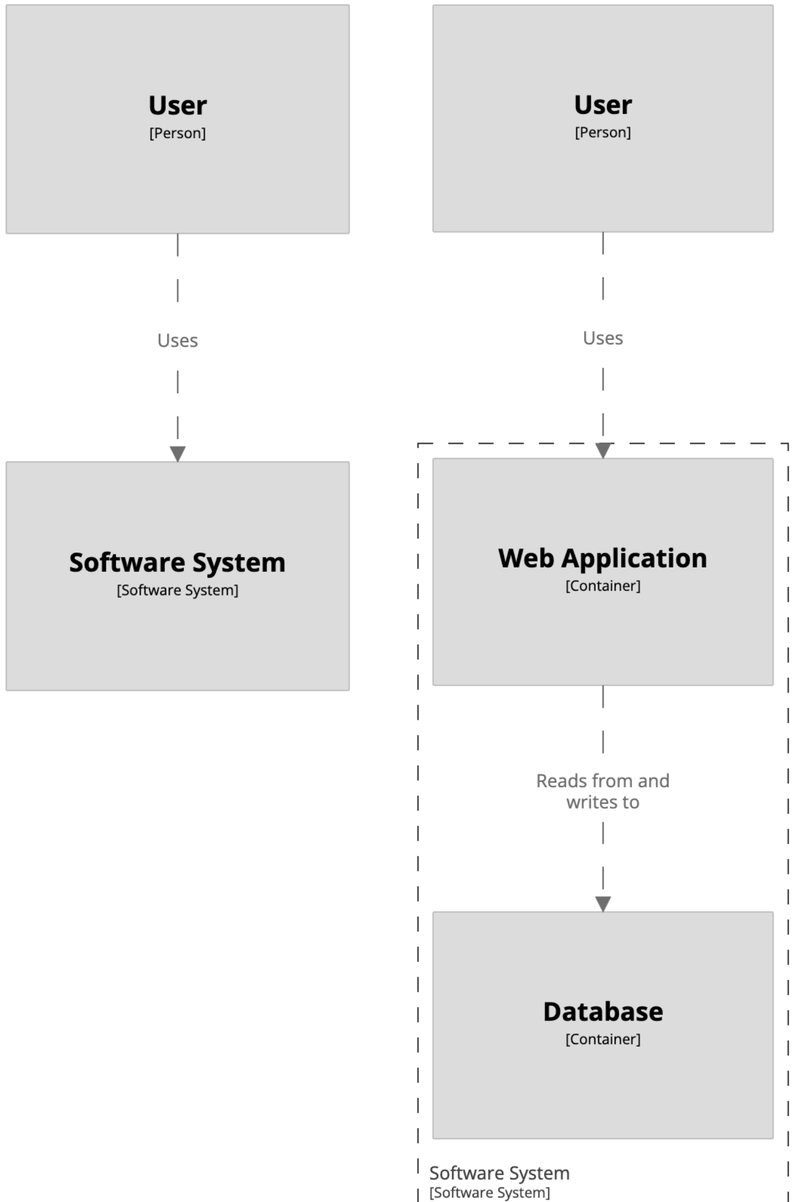


Unspecified relationships can be implied from the model

```
user -> softwareSystem "Uses"
```




```
user -> webapp "Uses"
webapp -> database "Reads from and writes to"
```

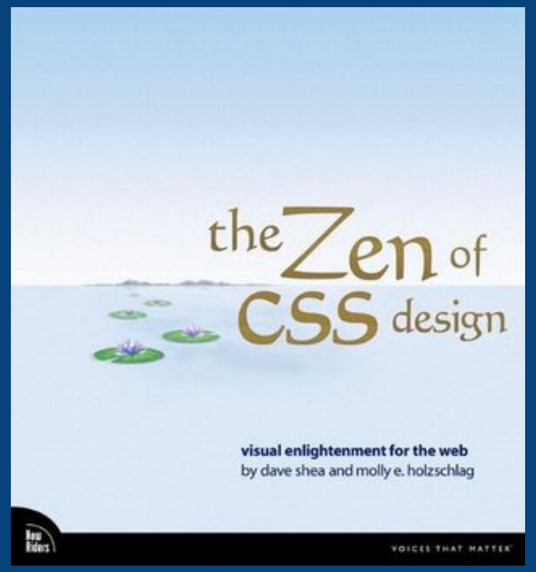


Implied relationships can be disabled using:

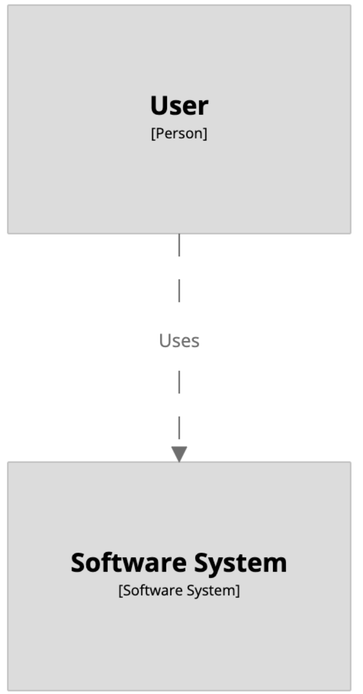
```
!impliedRelationships false
```

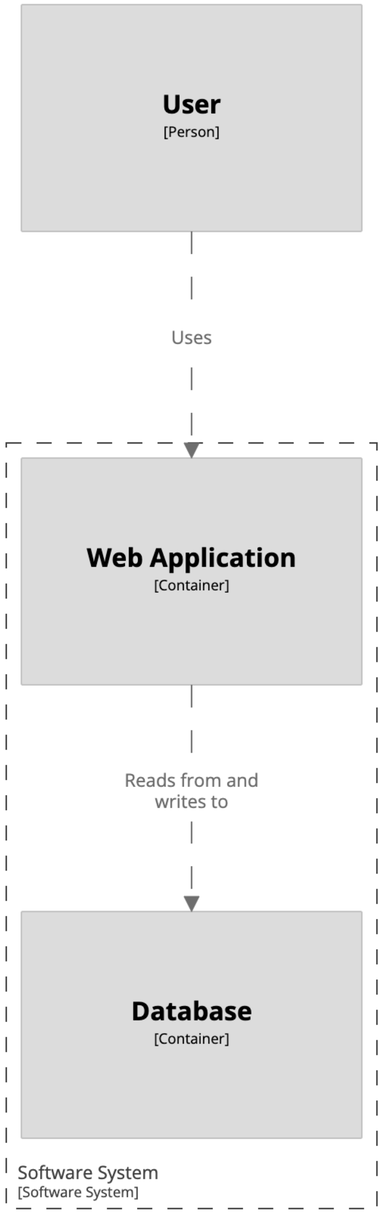
Separation of content and presentation

HTML & CSS

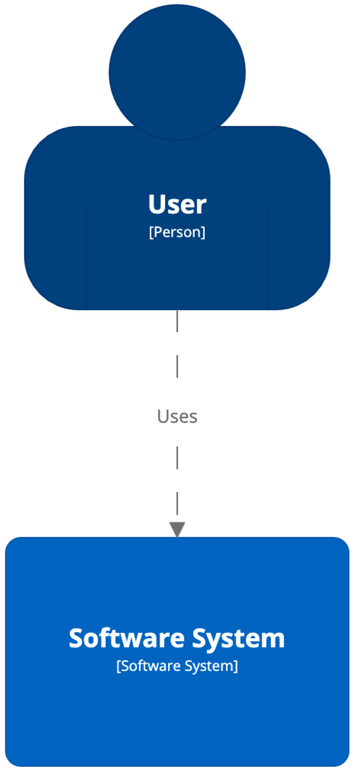

the Zen of CSS design
visual enlightenment for the web
by dave shea and molly e. holzschlag
New Riders
VOICES THAT MATTER

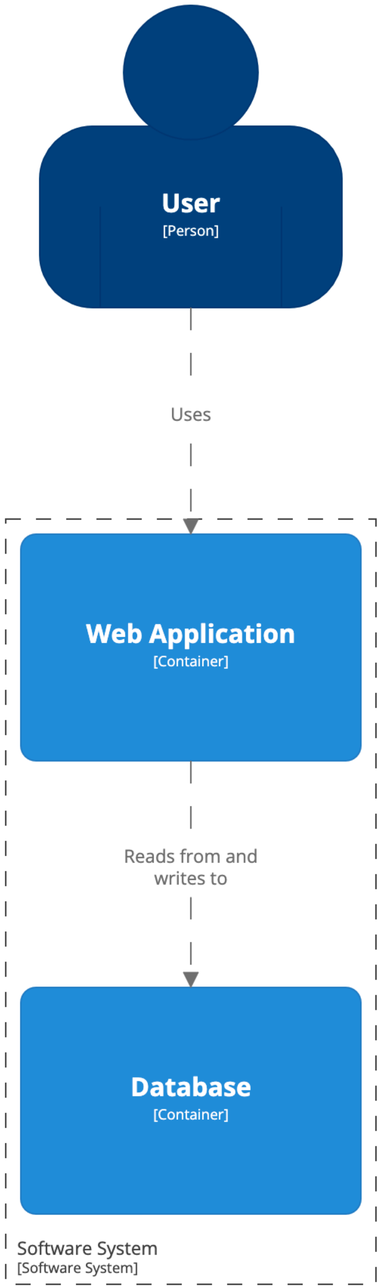
```
workspace {

    model {
        user = person "User"
        softwareSystem = softwareSystem "Software System" {
            webapp = container "Web Application"
            database = container "Database"
        }

        user -> webapp "Uses"
        webapp -> database "Reads from and writes to"
    }

    views {
        systemContext softwareSystem {
            include *
            autoLayout
        }

        container softwareSystem {
            include *
            autolayout
        }
    }

}
```

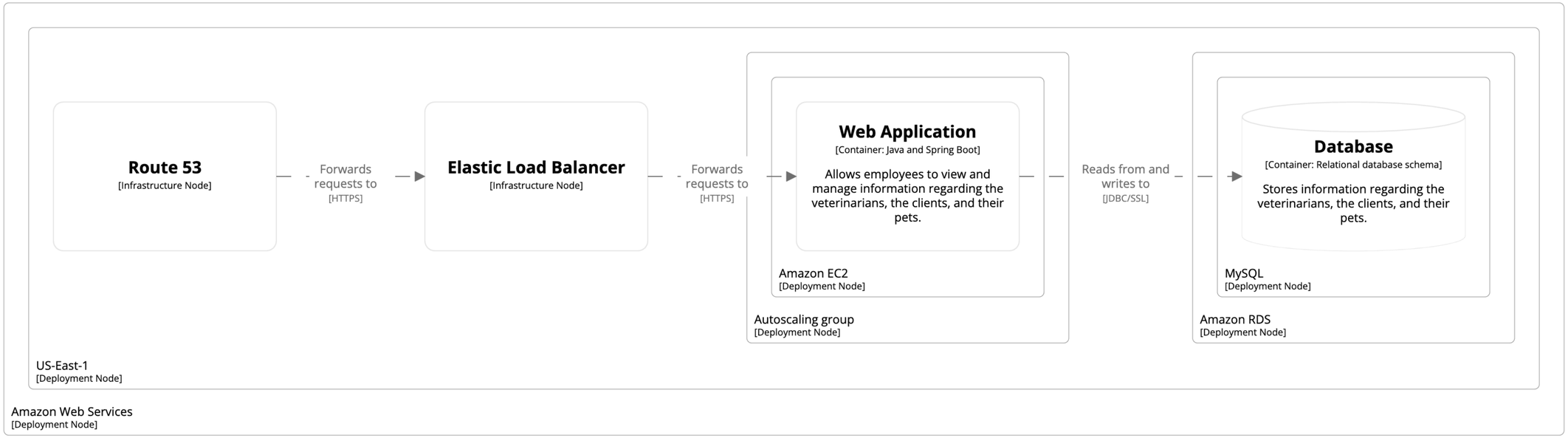




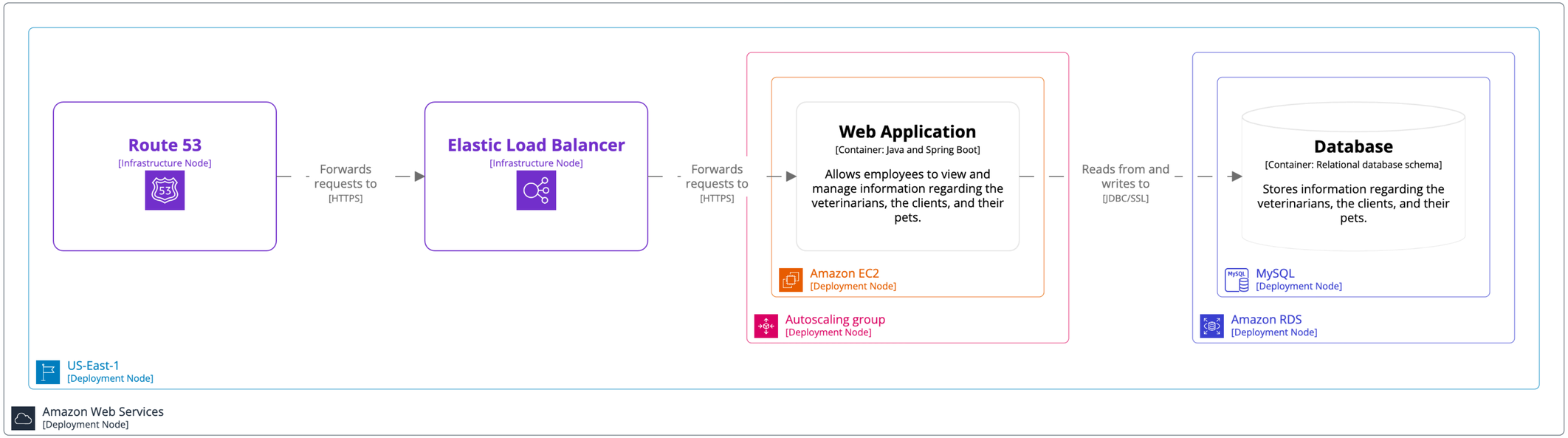
```
workspace {

    model {
        user = person "User"
        softwareSystem = softwareSystem "Software System" {
            webapp = container "Web Application"
            database = container "Database"
        }

        user -> webapp "Uses"
        webapp -> database "Reads from and writes to"
    }

    views {
        systemContext softwareSystem {
            include *
            autoLayout
        }

        container softwareSystem {
            include *
            autolayout
        }

        theme default
    }
```

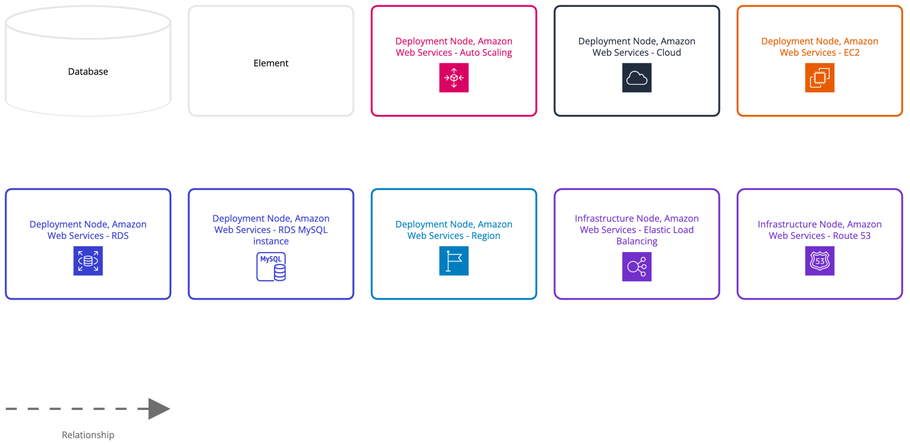





Route 53
[Infrastructure Node]
Forwards requests to
[HTTPS]
Elastic Load Balancer
[Infrastructure Node]
Forwards requests to
[HTTPS]
Web Application
[Container: Java and Spring Boot]
Allows employees to view and manage information regarding the veterinarians, the clients, and their pets.
Amazon EC2
[Deployment Node]
Autoscaling group
[Deployment Node]
Reads from and writes to
[JDBC/SSL]
Database
[Container: Relational database schema]
Stores information regarding the veterinarians, the clients, and their pets.
MySQL
[Deployment Node]
Amazon RDS
[Deployment Node]
US-East-1
[Deployment Node]
Amazon Web Services
[Deployment Node]


Database
Element
Infrastructure Node
Relationship

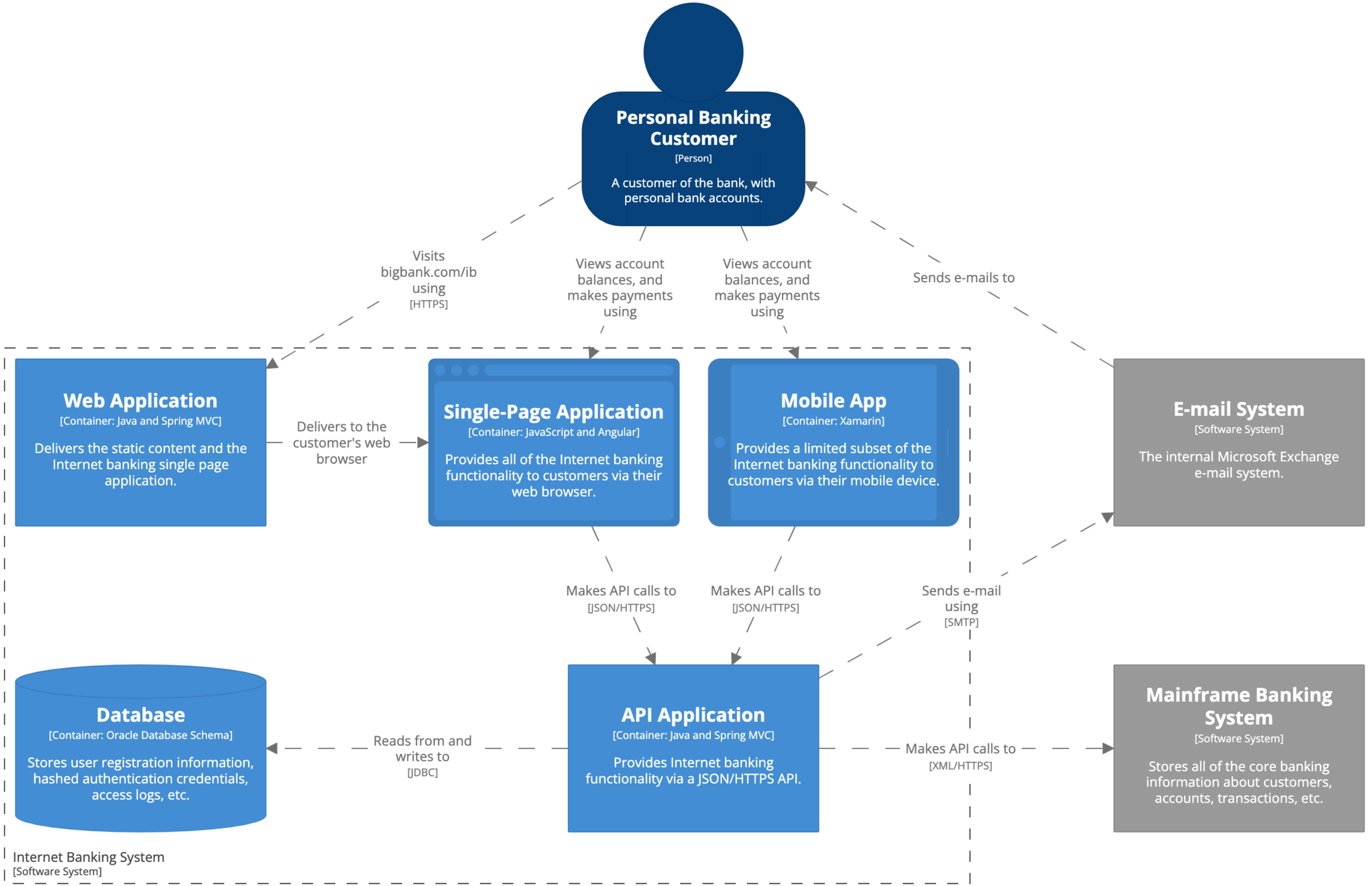

Route 53
[Infrastructure Node]
Forwards requests to
[HTTPS]
Elastic Load Balancer
[Infrastructure Node]
Forwards requests to
[HTTPS]
Web Application
[Container: Java and Spring Boot]
Allows employees to view and manage information regarding the veterinarians, the clients, and their pets.
Amazon EC2
[Deployment Node]
Autoscaling group
[Deployment Node]
Reads from and writes to
[JDBC/SSL]
Database
[Container: Relational database schema]
Stores information regarding the veterinarians, the clients, and their pets.
MySQL
[Deployment Node]
Amazon RDS
[Deployment Node]
US-East-1
[Deployment Node]
Amazon Web Services
[Deployment Node]

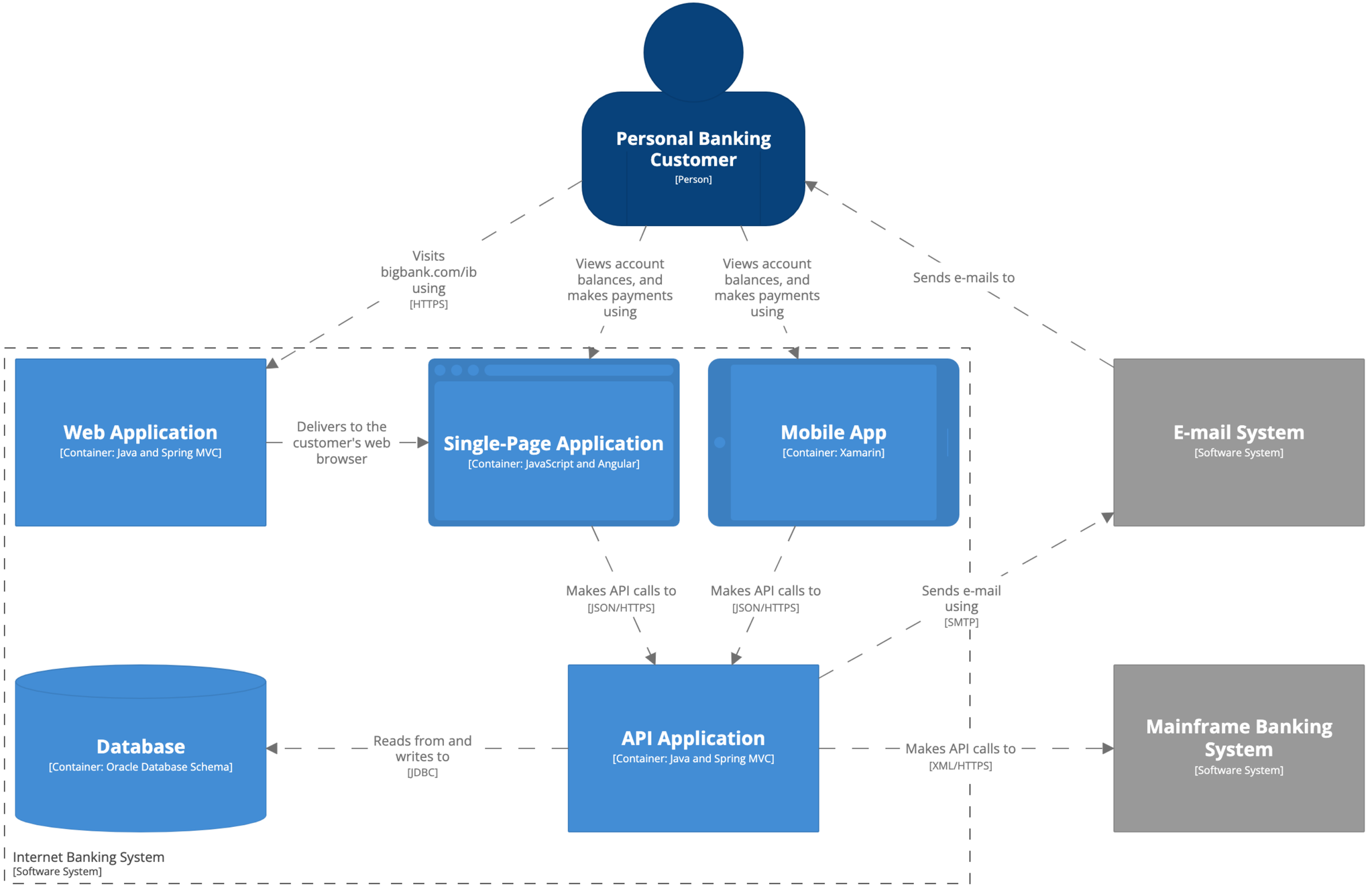

Database
Element
Deployment Node, Amazon Web Services - Auto Scaling
Deployment Node, Amazon Web Services - Cloud
Deployment Node, Amazon Web Services - EC2
Deployment Node, Amazon Web Services - RDS
Deployment Node, Amazon Web Services - RDS MySQL instance
Deployment Node, Amazon Web Services - Region
Infrastructure Node, Amazon Web Services - Elastic Load Balancing
Infrastructure Node, Amazon Web Services - Route 53
Relationship

Styling of elements and relationships is achieved via tags

```
workspace {

    model {
        softwareSystem "Software System"
    }

    views {
        systemLandscape {
            include *
            autolayout
        }









    }

}
```

```
workspace {

    model {
        softwareSystem "Software System"
    }

    views {
        systemLandscape {
            include *
            autolayout
        }

        styles {
            element "Software System" {
                background #1168bd
                color #ffffff
                shape RoundedBox
            }
        }
    }

}
```


Personal Banking Customer
[Person]
A customer of the bank, with personal bank accounts.
Visits bigbank.com/ib using
[HTTPS]
Views account balances, and makes payments using
Views account balances, and makes payments using
Sends e-mails to
Web Application
[Container: Java and Spring MVC]
Delivers the static content and the Internet banking single page application.
Delivers to the customer's web browser
Single-Page Application
[Container: JavaScript and Angular]
Provides all of the Internet banking functionality to customers via their web browser.
Mobile App
[Container: Xamarin]
Provides a limited subset of the Internet banking functionality to customers via their mobile device.
E-mail System
[Software System]
The internal Microsoft Exchange e-mail system.
Makes API calls to
[JSON/HTTPS]
Makes API calls to
[JSON/HTTPS]
Sends e-mail using
[SMTP]
Database
[Container: Oracle Database Schema]
Stores user registration information, hashed authentication credentials, access logs, etc.
Reads from and writes to
[JDBC]
API Application
[Container: Java and Spring MVC]
Provides Internet banking functionality via a JSON/HTTPS API.
Makes API calls to
[XML/HTTPS]
Mainframe Banking System
[Software System]
Stores all of the core banking information about customers, accounts, transactions, etc.
Internet Banking System
[Software System]


Personal Banking Customer
[Person]
Visits bigbank.com/ib using
[HTTPS]
Views account balances, and makes payments using
Views account balances, and makes payments using
Sends e-mails to
Web Application
[Container: Java and Spring MVC]
Delivers to the customer's web browser
Single-Page Application
[Container: JavaScript and Angular]
Mobile App
[Container: Xamarin]
E-mail System
[Software System]
Makes API calls to
[JSON/HTTPS]
Makes API calls to
[JSON/HTTPS]
Sends e-mail using
[SMTP]
Database
[Container: Oracle Database Schema]
Reads from and writes to
[JDBC]
API Application
[Container: Java and Spring MVC]
Makes API calls to
[XML/HTTPS]
Mainframe Banking System
[Software System]
Internet Banking System
[Software System]

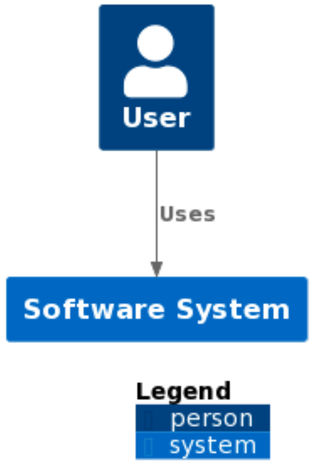
Rendering tool independent

“Diagrams as code 1.0”

PlantUML, Mermaid, etc are **input formats**

Diagrams as code 1.0

```
@startuml
!include https://raw.githubusercontent.com/plantuml-stdlib/C4-PlantUML/master/C4.puml
!include https://raw.githubusercontent.com/plantuml-stdlib/C4-PlantUML/master/C4_Context.puml

Person(User, "User")
System(SoftwareSystem, "Software System")

Rel_D(User, SoftwareSystem, "Uses")

SHOW_LEGEND()
@enduml
```

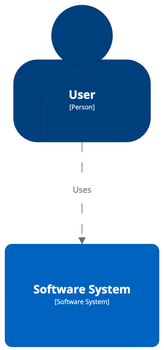


“Diagrams as code 2.0”

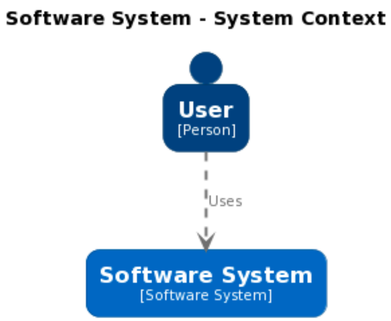
PlantUML, Mermaid, etc are **output formats**

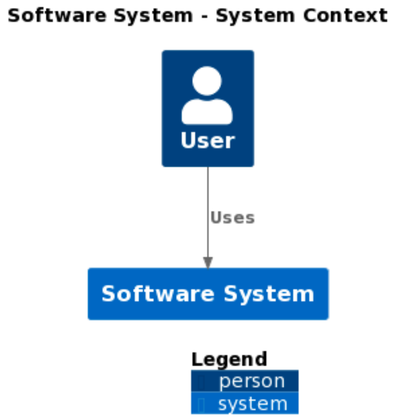
Diagrams as code 2.0

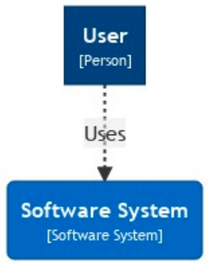
```
workspace {

    model {
        user = person "User"
        softwareSystem = softwareSystem "Software System"

        user -> softwareSystem "Uses"
    }

    views {
        theme default
    }

}
```

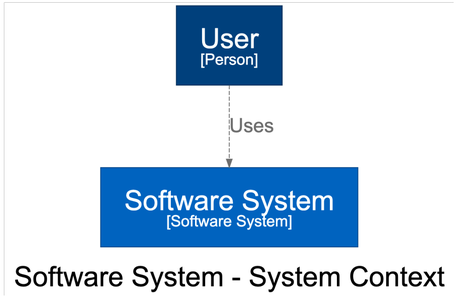


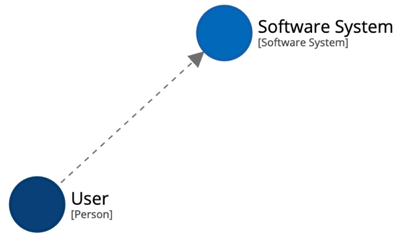
[System Context] Software System
Friday, 4 March 2022, 09:27 Greenwich Mean Time

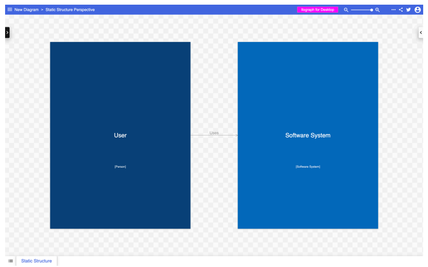












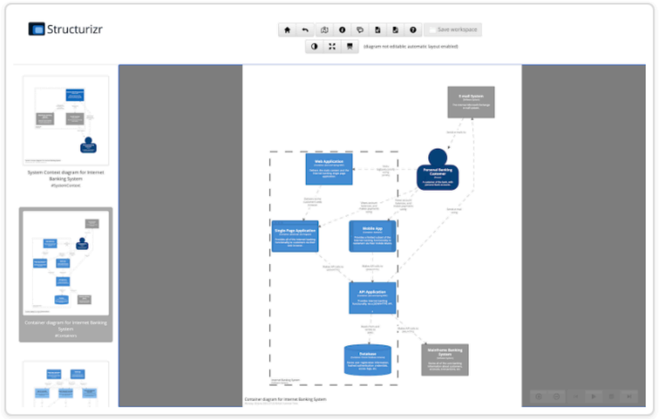
Automatic layout
vs
manual layout?

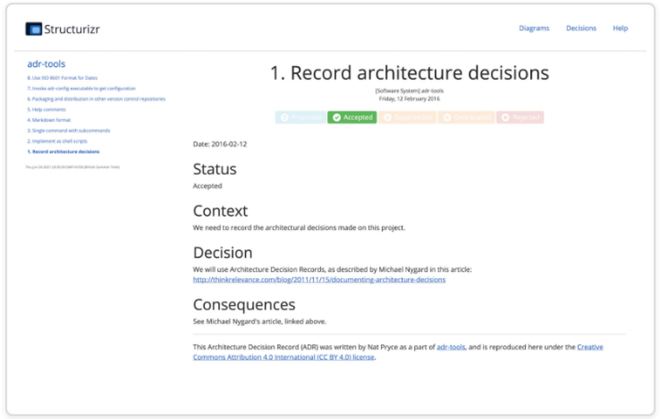
Structurizr Lite

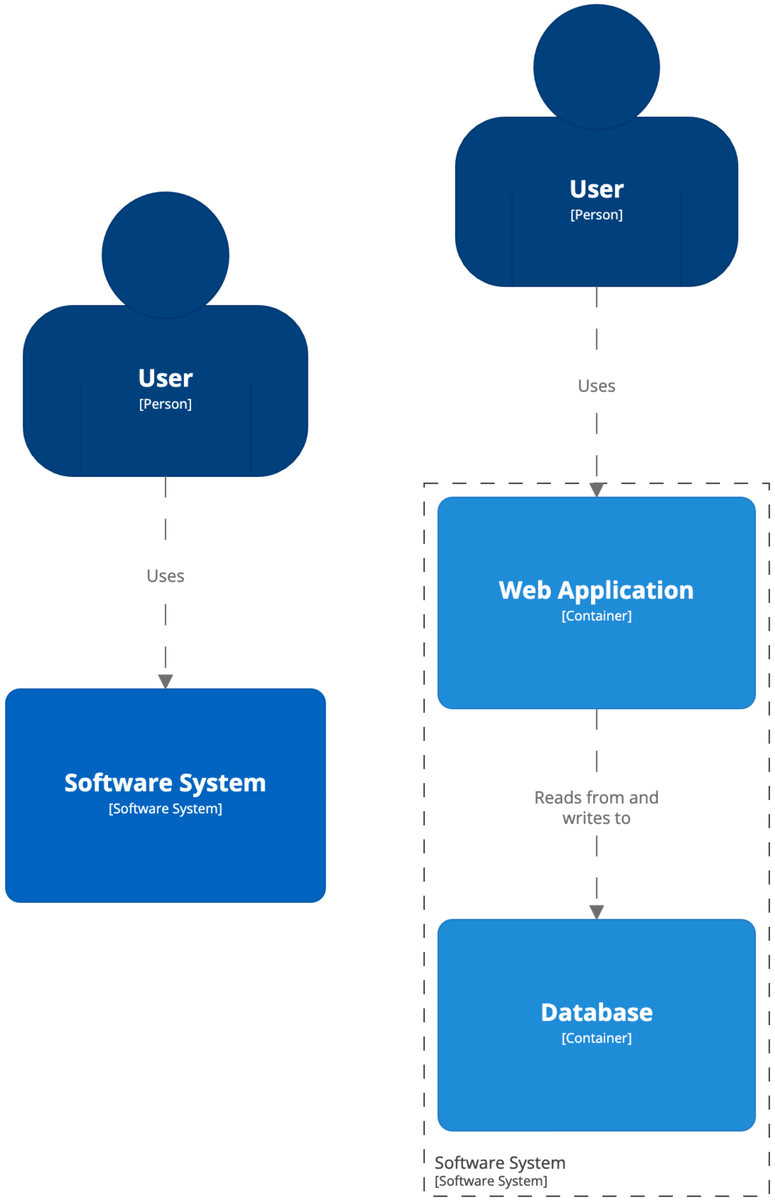
Overview | Getting started | Usage | Auto-sync | Workflow | Docker Hub | EULA

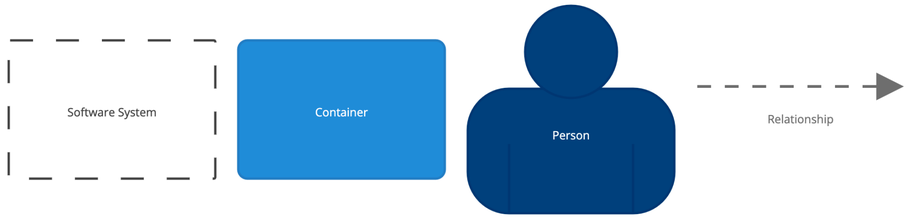
Overview

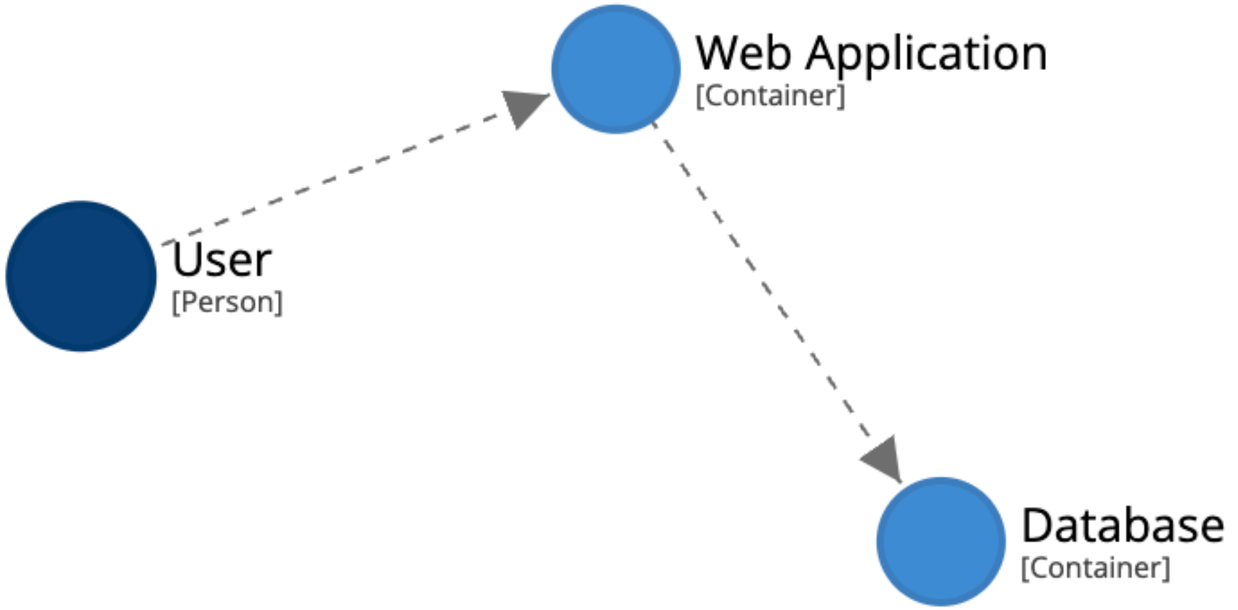
Packaged as a Docker container, and designed for developers, this version of Structurizr provides a way to quickly work with a single workspace. It's free to use, and allows you to view/edit diagrams, view documentation, and view architecture decision records defined in a DSL or JSON workspace.






User
[Person]
Uses
Software System
[Software System]
User
[Person]
Uses
Web Application
[Container]
Reads from and
writes to
Database
[Container]
Software System
[Software System]


Software System
Container
Person
Relationship

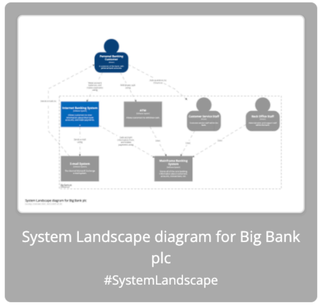

Web Application
[Container]
User
[Person]
Database
[Container]

```
docker run -it --rm -p 8080:8080 -v /Users/simon/bigbankplc/:/usr/local/structurizr structurizr/lite
```

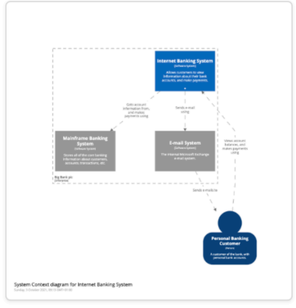
Structurizr

Save workspace

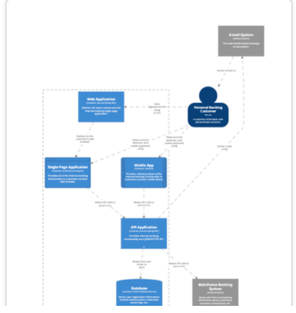
(diagram not editable; automatic layout enabled)

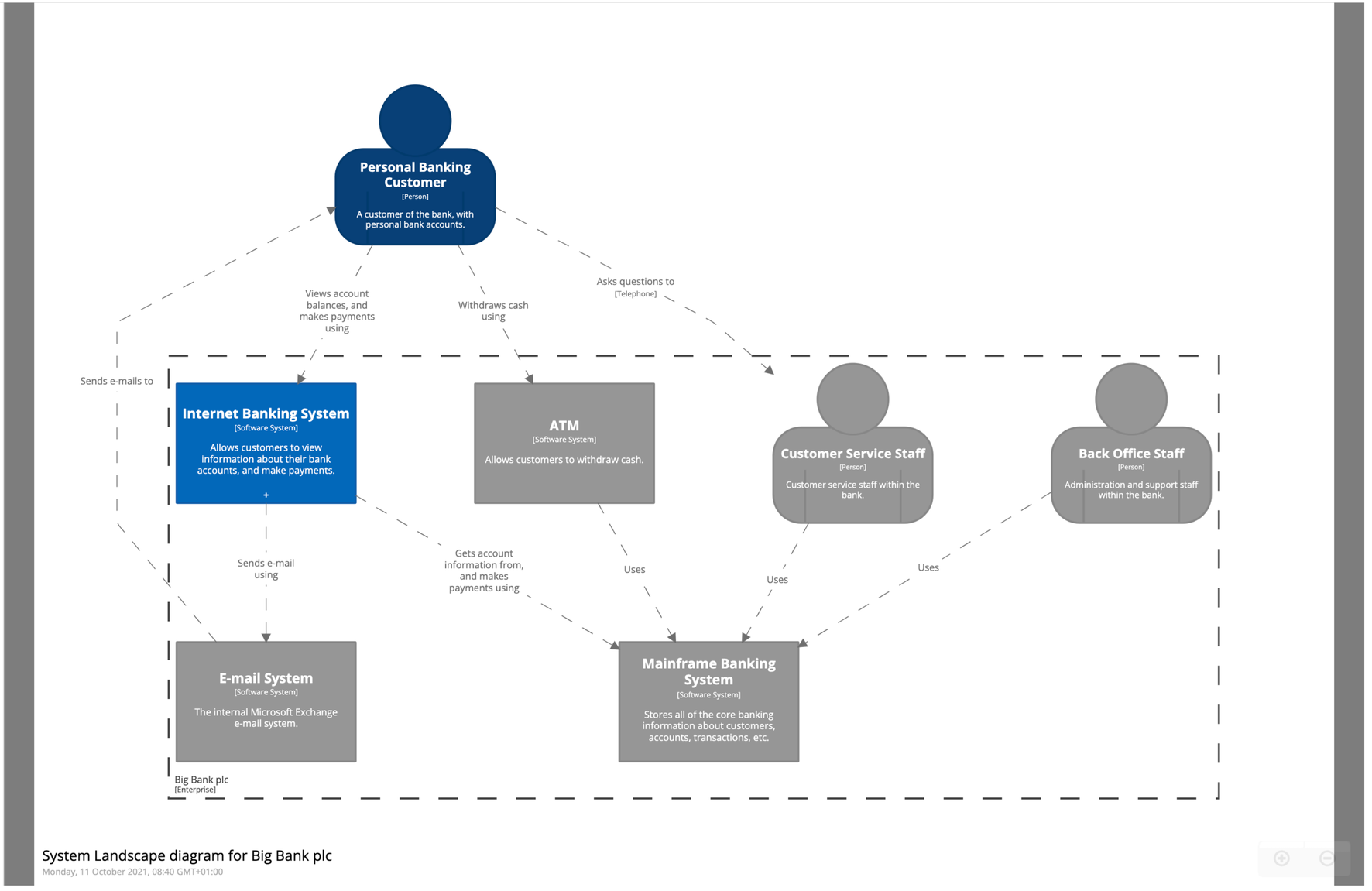


System Landscape diagram for Big Bank plc
#SystemLandscape



System Context diagram for Internet Banking System
#SystemContext





System Landscape diagram for Big Bank plc
Monday, 11 October 2021, 08:40 GMT+01:00

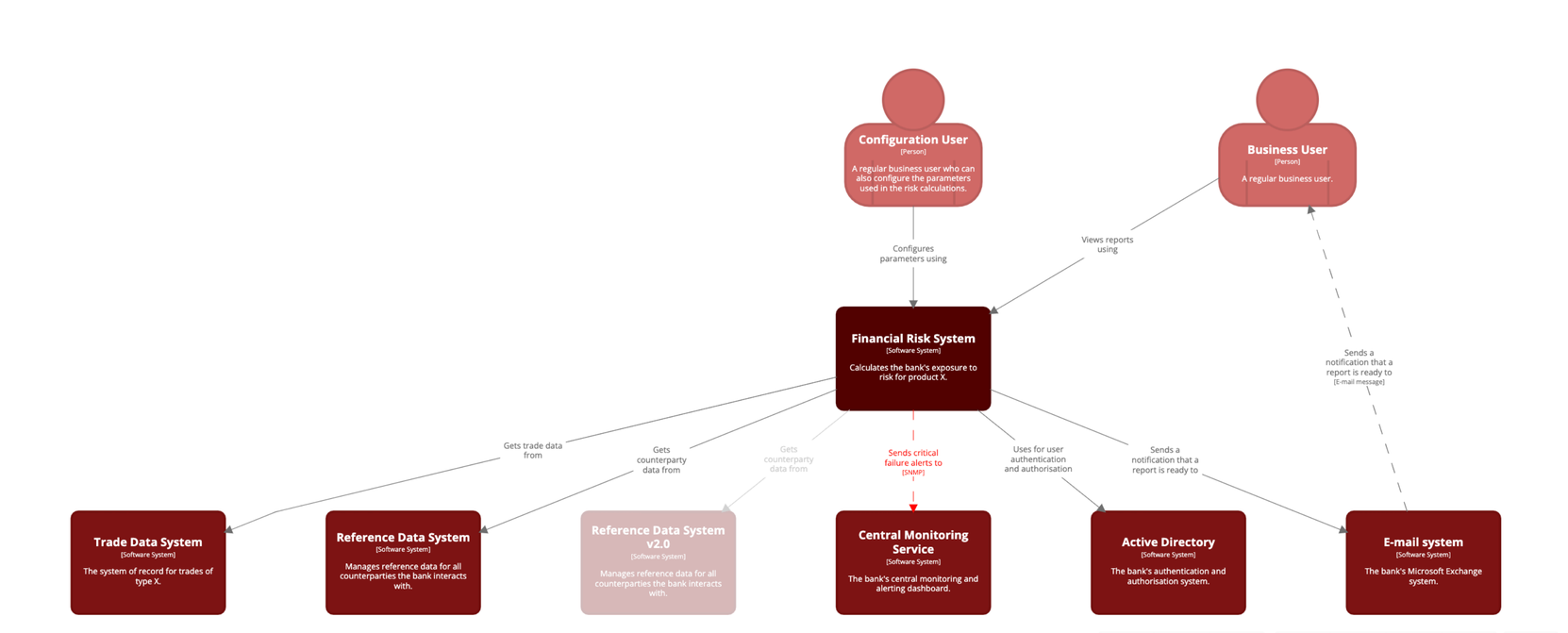
!docs <directory name>

Monday, 11 October 2021, 08:45 GMT+01:00

Financial Risk System

1 Context

A global investment bank based in London, New York and Singapore trades (buys and sells) financial products with other banks (counterparties). When share prices on the stock markets move up or down, the bank either makes money or loses it. At the end of the working day, the bank needs to gain a view of how much risk they are exposed to (e.g. of losing money) by running some calculations on the data held about their trades. The bank has an existing Trade Data System (TDS) and Reference Data System (RDS) but need a new Risk System.



System Context diagram for Financial Risk System

An example System Context diagram for the Financial Risk System architecture kata.
Monday, 11 October 2021, 08:45 GMT+01:00

1.1 Trade Data System

!adrs <directory name>

Structurizr

Diagrams Documentation Decisions Help

adr-tools

Sun Oct 03 2021 10:08:10 GMT+0100 (GMT+01:00)

adr-tools - Summary

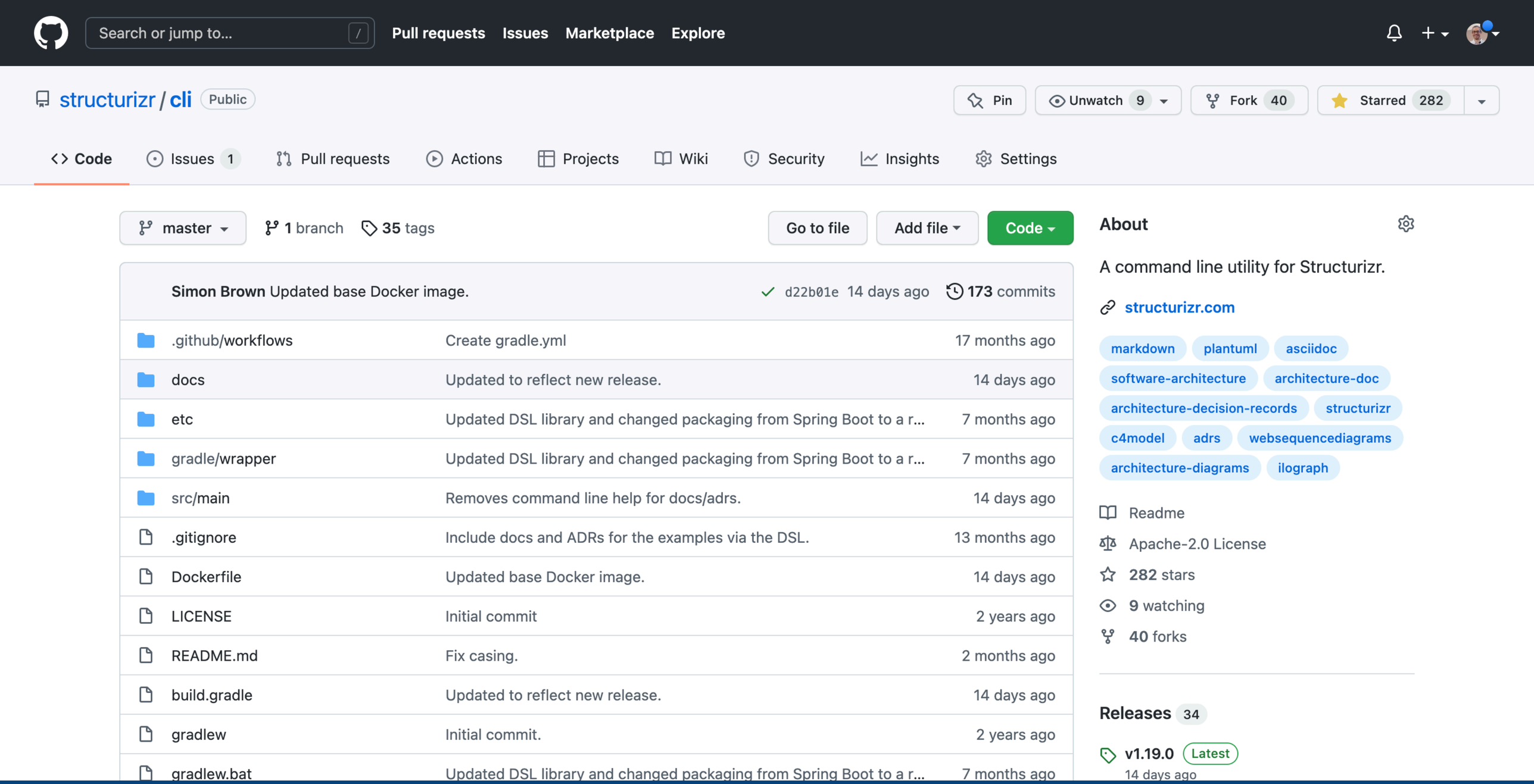
[Software System] adr-tools

2017

8. Use ISO 8601 Format for Dates Tuesday, 21 February 2017	Accepted

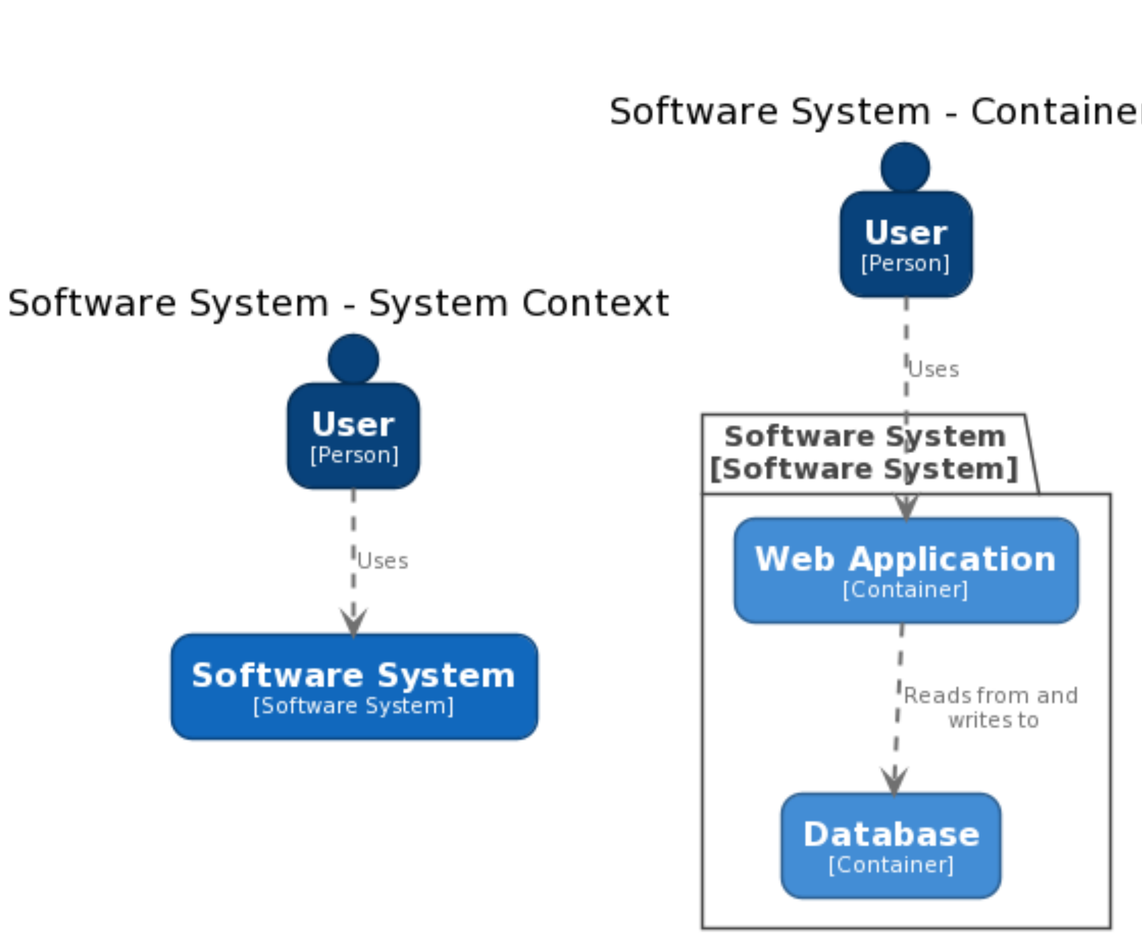
2016

7. Invoke adr-config executable to get configuration Saturday, 17 December 2016	Accepted
6. Packaging and distribution in other version control repositories Tuesday, 16 February 2016	Accepted
5. Help comments Saturday, 13 February 2016	Accepted
4. Markdown format Friday, 12 February 2016	Accepted
3. Single command with subcommands Friday, 12 February 2016	Accepted
2. Implement as shell scripts Friday, 12 February 2016	Accepted
1. Record architecture decisions Friday, 12 February 2016	Accepted

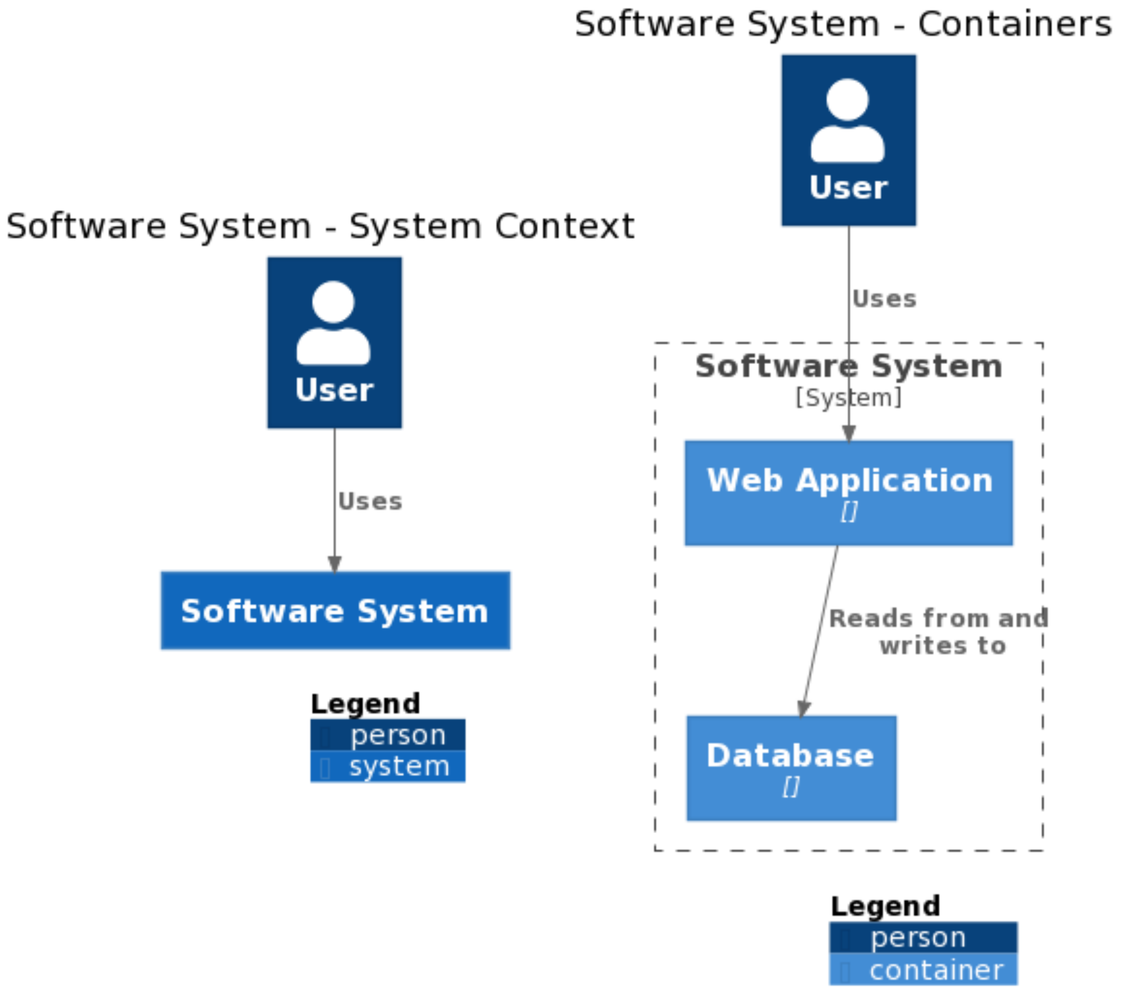


https://github.com/structurizr/cli

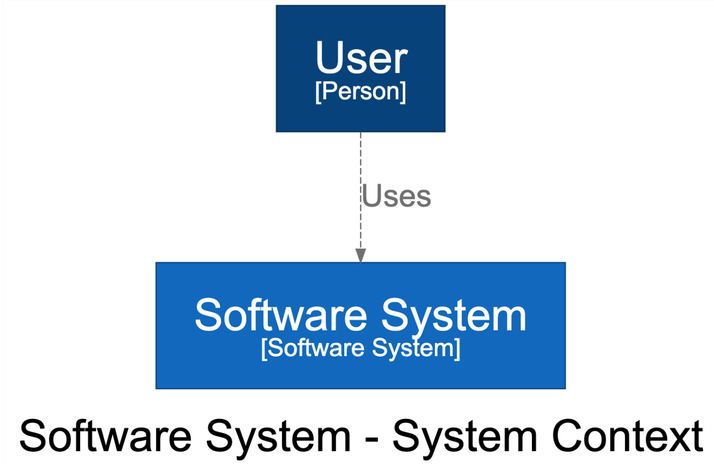
```
./structurizr.sh export -workspace /Users/simon/bigbankplc/workspace.dsl -format plantuml

Exporting workspace from /Users/simon/bigbankplc/workspace.dsl
 - loading workspace from DSL
 - using StructurizrPlantUMLExporter
 - writing /Users/simon/bigbankplc/structurizr-SystemLandscape.puml
 - writing /Users/simon/bigbankplc/structurizr-SystemContext.puml
 - writing /Users/simon/bigbankplc/structurizr-Containers.puml
 - writing /Users/simon/bigbankplc/structurizr-Components.puml
 - writing /Users/simon/bigbankplc/structurizr-SignIn.puml
 - writing /Users/simon/bigbankplc/structurizr-LiveDeployment.puml
 - writing /Users/simon/bigbankplc/structurizr-DevelopmentDeployment.puml
 - writing /Users/simon/bigbankplc/structurizr-SignIn-sequence.puml
 - finished
```



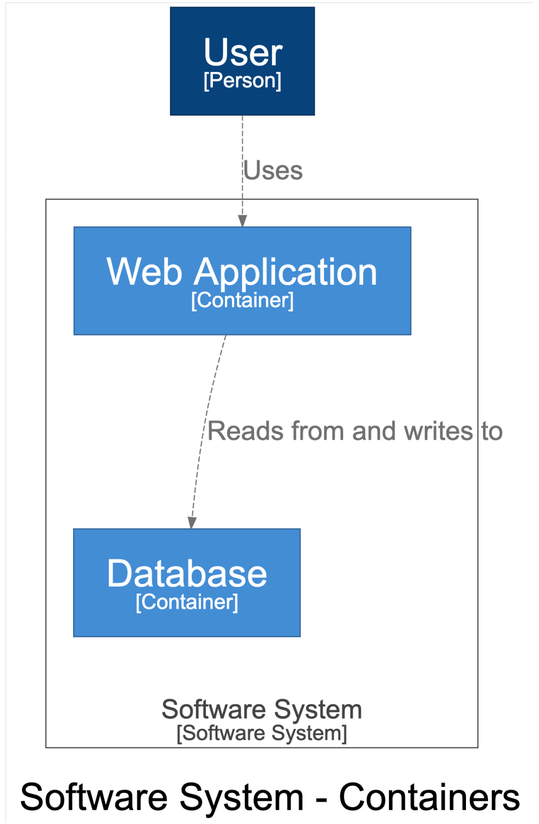
PlantUML



C4-PlantUML

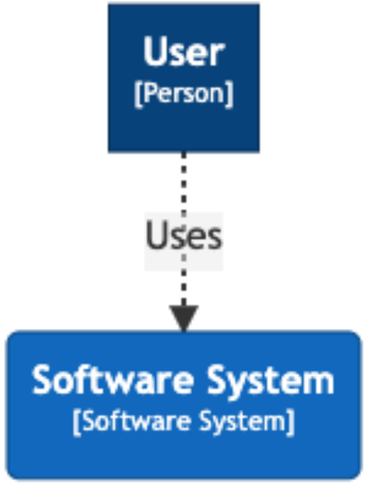


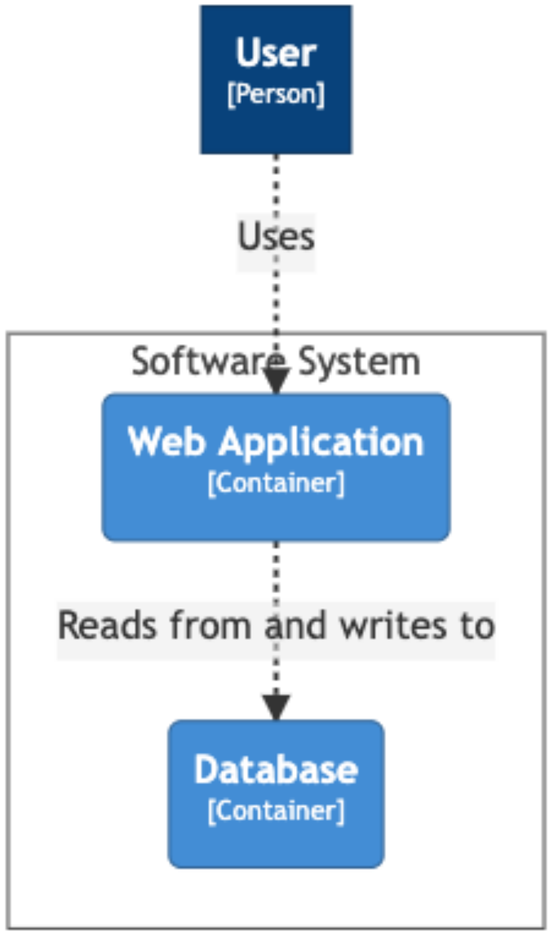
Software System - System Context



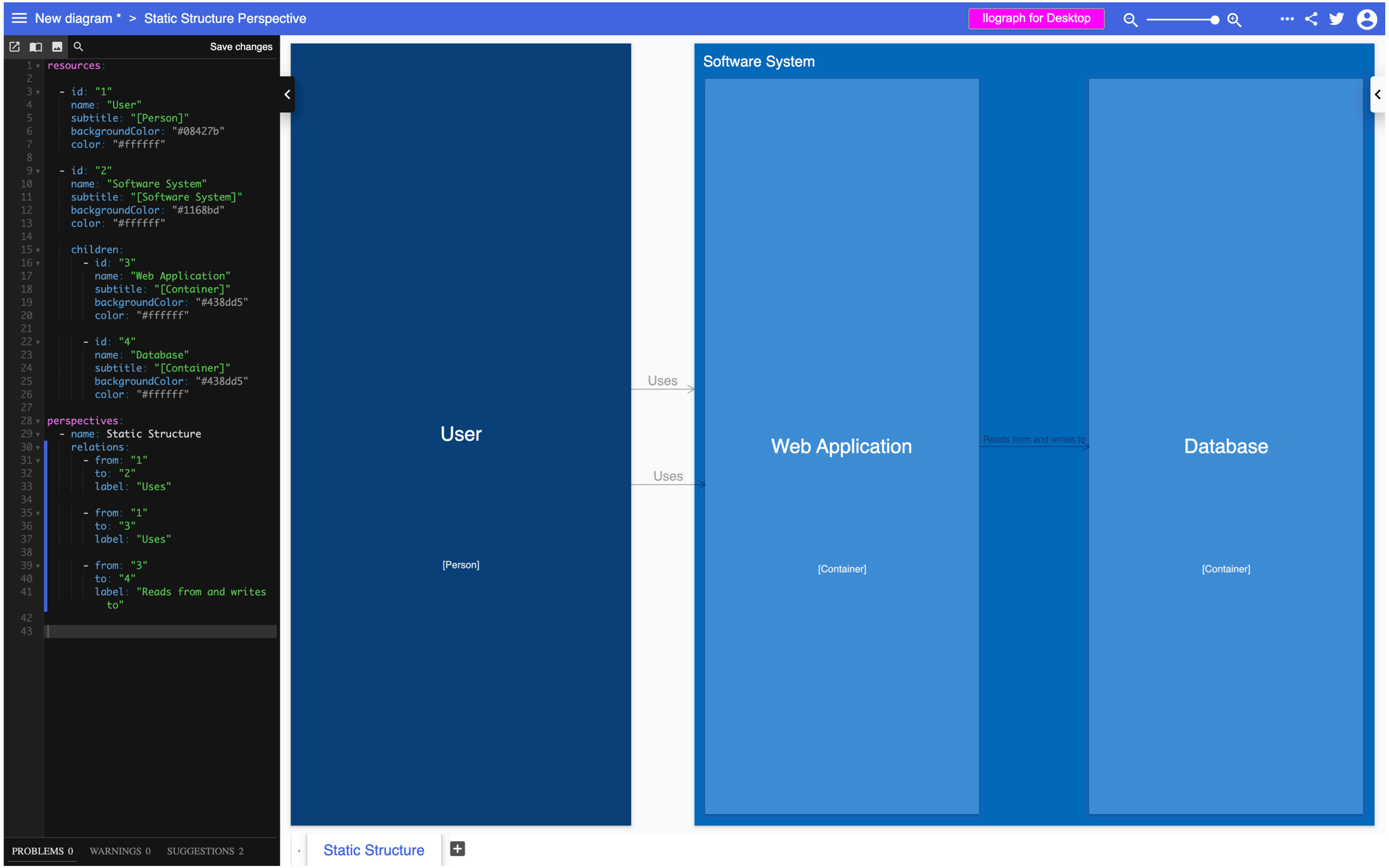
Software System - Containers

Graphviz/DOT

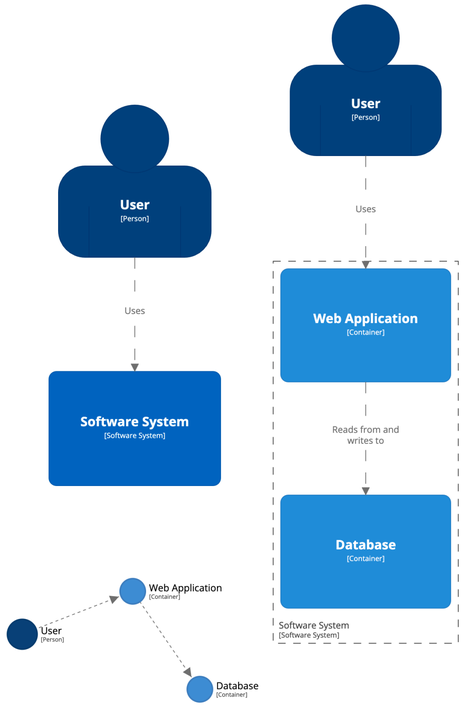


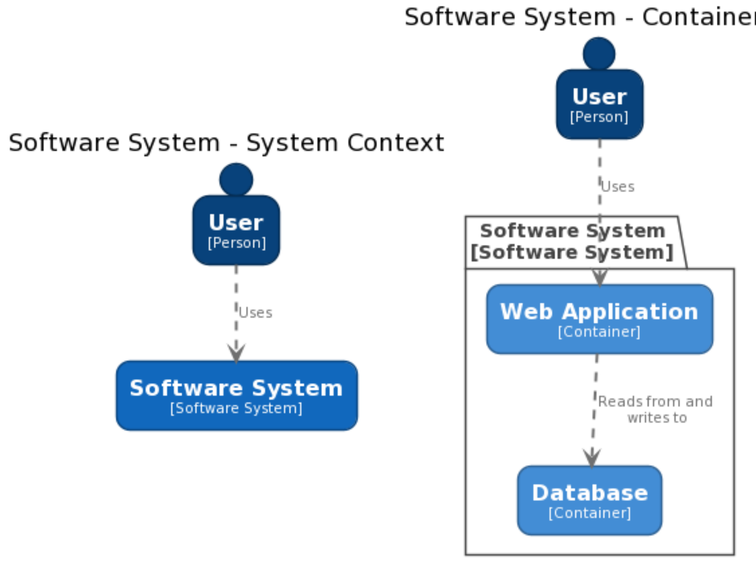


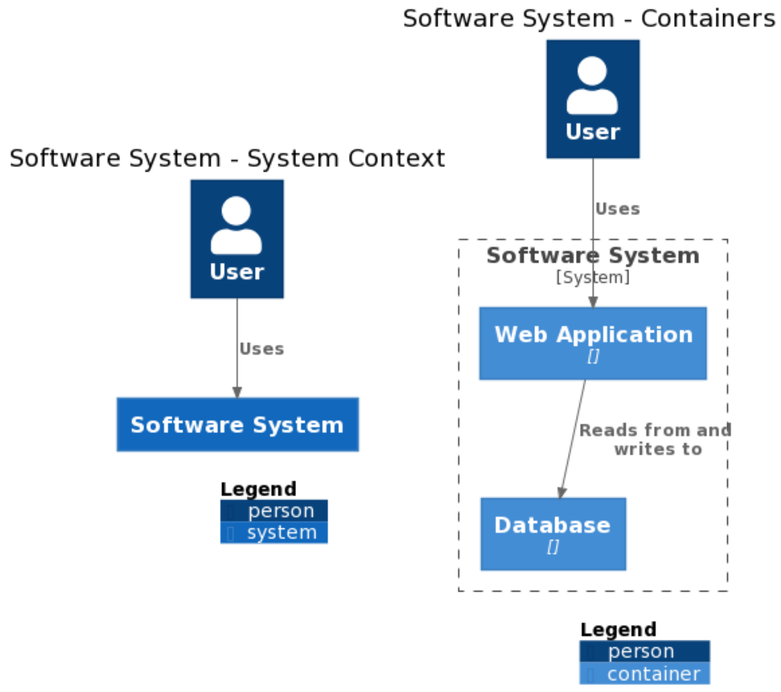
Mermaid

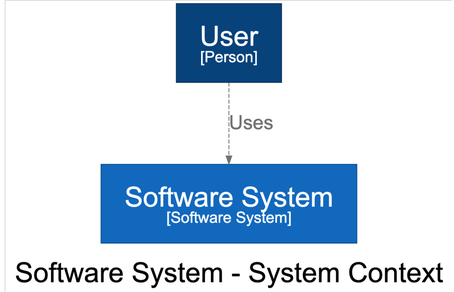

New diagram * > Static Structure Perspective
Ilograph for Desktop
Save changes
resources:
- id: "1"
name: "User"
subtitle: "[Person]"
backgroundColor: "#08427b"
color: "#ffffff"
- id: "2"
name: "Software System"
subtitle: "[Software System]"
backgroundColor: "#1168bd"
color: "#ffffff"
children:
- id: "3"
name: "Web Application"
subtitle: "[Container]"
backgroundColor: "#438dd5"
color: "#ffffff"
- id: "4"
name: "Database"
subtitle: "[Container]"
backgroundColor: "#438dd5"
color: "#ffffff"
perspectives:
- name: Static Structure
relations:
- from: "1"
to: "2"
label: "Uses"
- from: "1"
to: "3"
label: "Uses"
- from: "3"
to: "4"
label: "Reads from and writes to"
PROBLEMS 0
WARNINGS 0
SUGGESTIONS 2
User
[Person]
Uses
Uses
Software System
Web Application
[Container]
Reads from and writes to
Database
[Container]
Static Structure

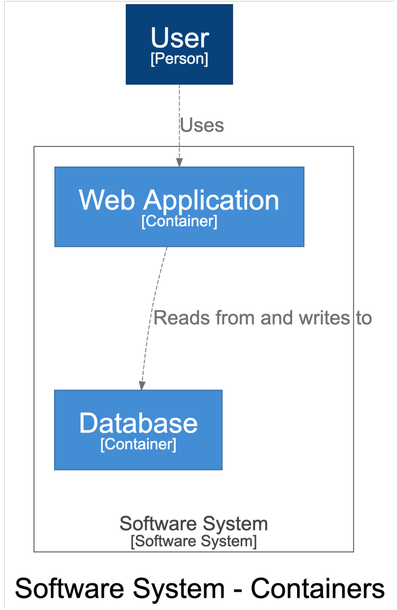
Ilograph

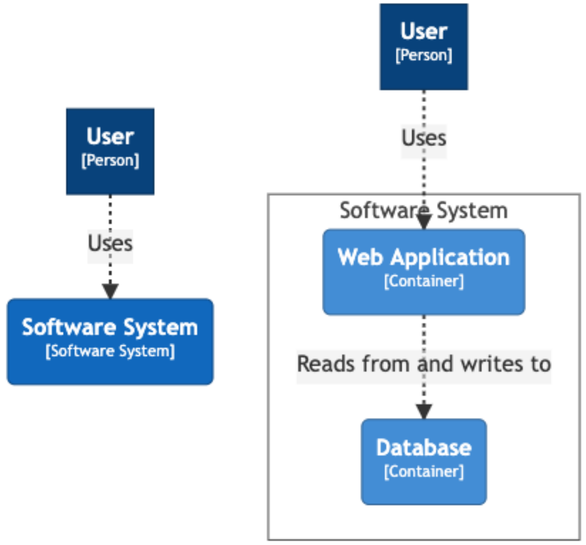

User
[Person]
Uses
Software System
[Software System]
User
[Person]
Uses
Web Application
[Container]
Reads from and
writes to
Database
[Container]
Software System
[Software System]
Web Application
[Container]
User
[Person]
Database
[Container]

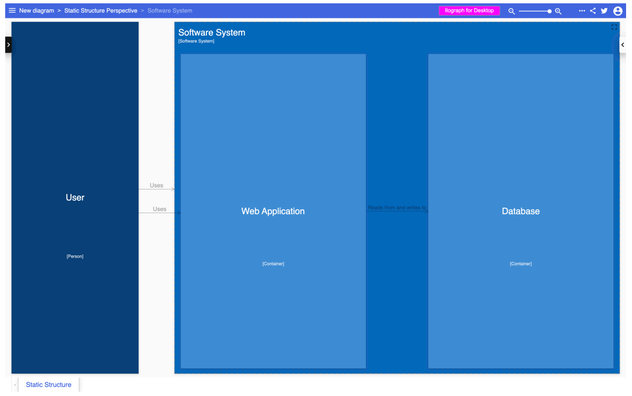

Software System - Containers
Software System - System Context
User
[Person]
Uses
Software System
[Software System]
User
[Person]
Uses
Software System
[Software System]
Web Application
[Container]
Reads from and
writes to
Database
[Container]

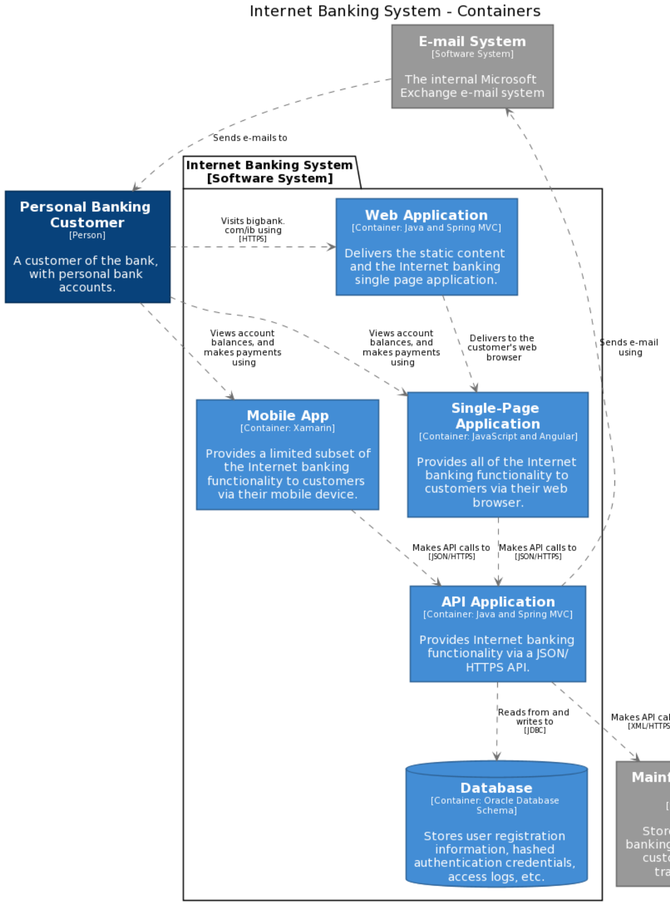

Software System - Containers
Software System - System Context
User
Uses
Software System
Legend
person
system
User
Uses
Software System
[System]
Web Application
[]
Reads from and
writes to
Database
[]
Legend
person
container

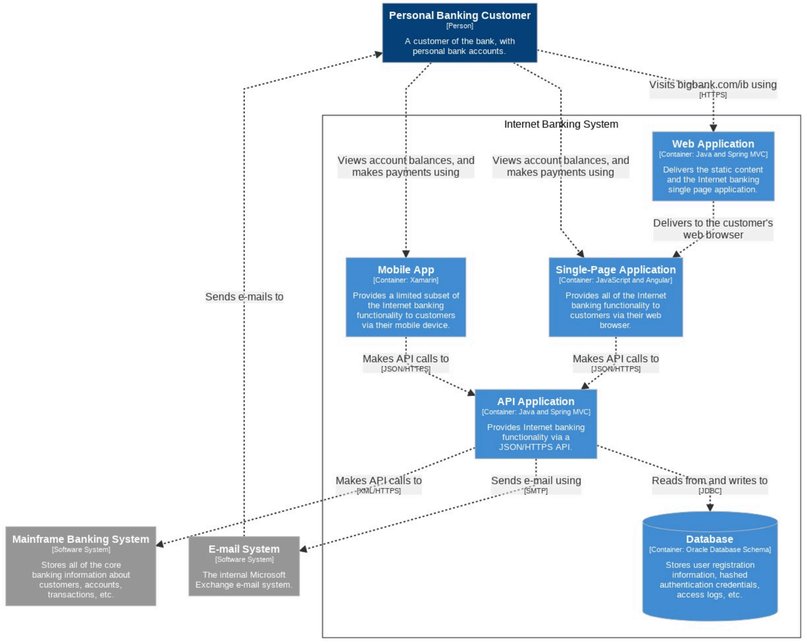

User
[Person]
Uses
Software System
[Software System]
Software System - System Context

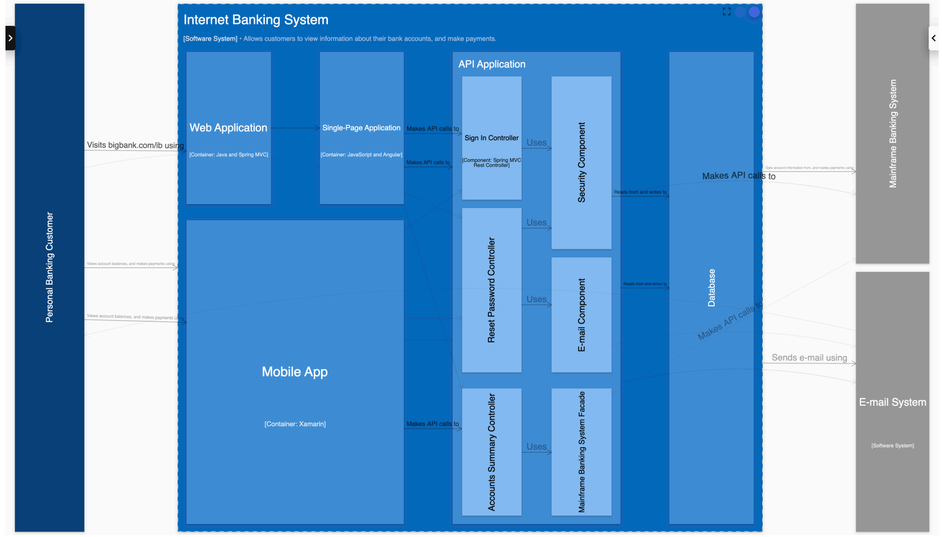

User
[Person]
Uses
Web Application
[Container]
Reads from and writes to
Database
[Container]
Software System
[Software System]
Software System - Containers

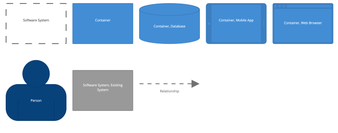

User
[Person]
Uses
Software System
[Software System]
User
[Person]
Uses
Software System
Web Application
[Container]
Reads from and writes to
Database
[Container]

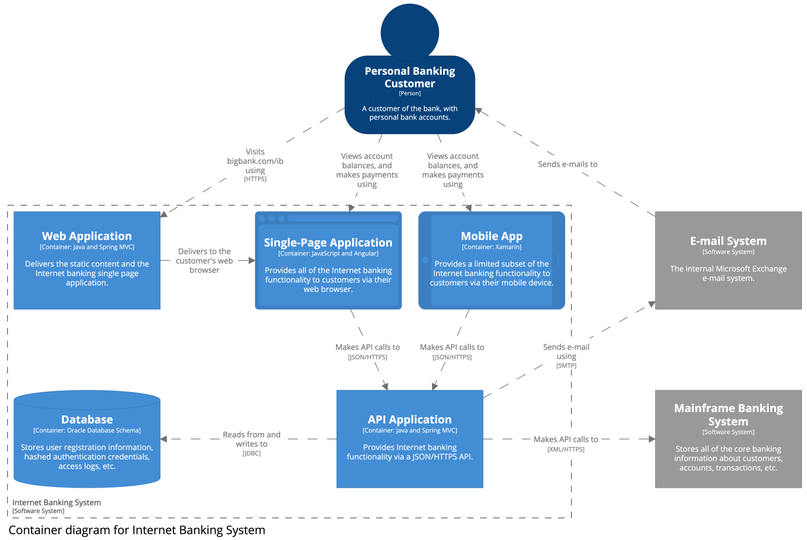

New diagram > Static Structure Perspective > Software System
Signup for Desktop
User
[Person]
Uses
Uses
Software System
[Software System]
Web Application
[Container]
Reads from and writes to
Database
[Container]
Static Structure





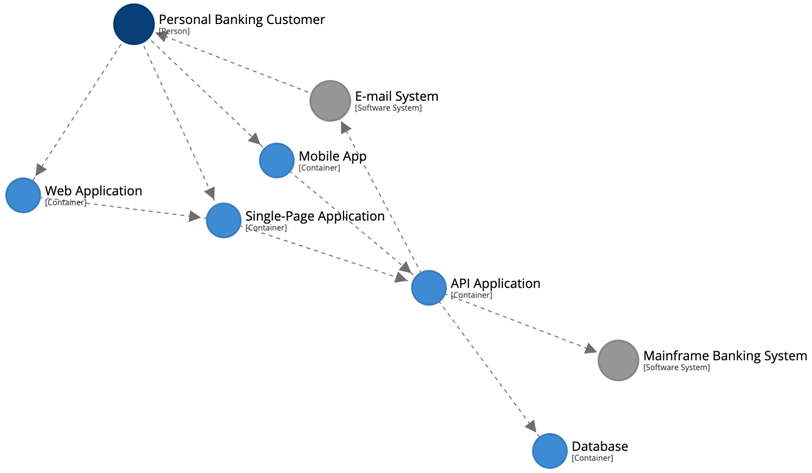






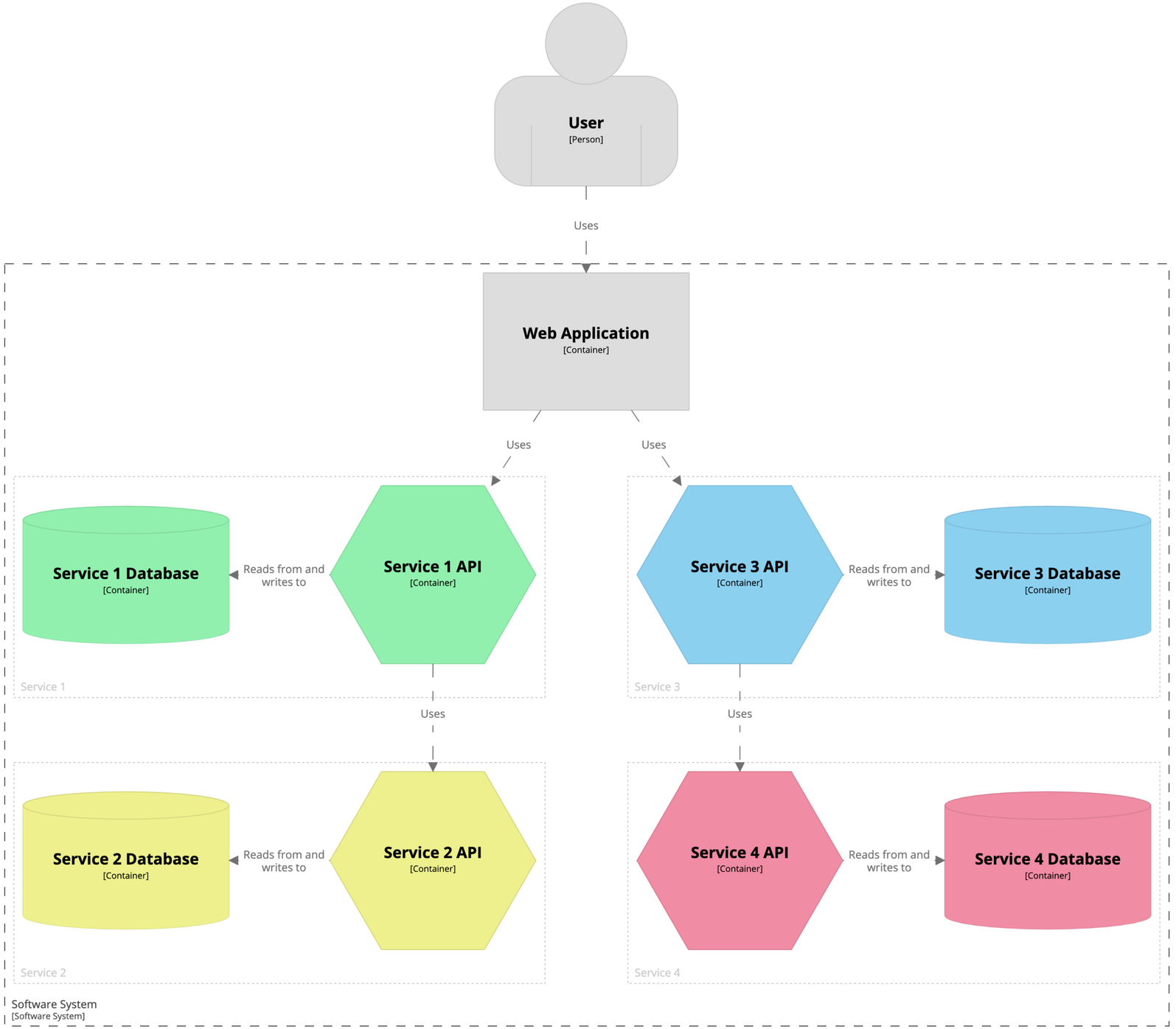
Container diagram for Internet Banking System

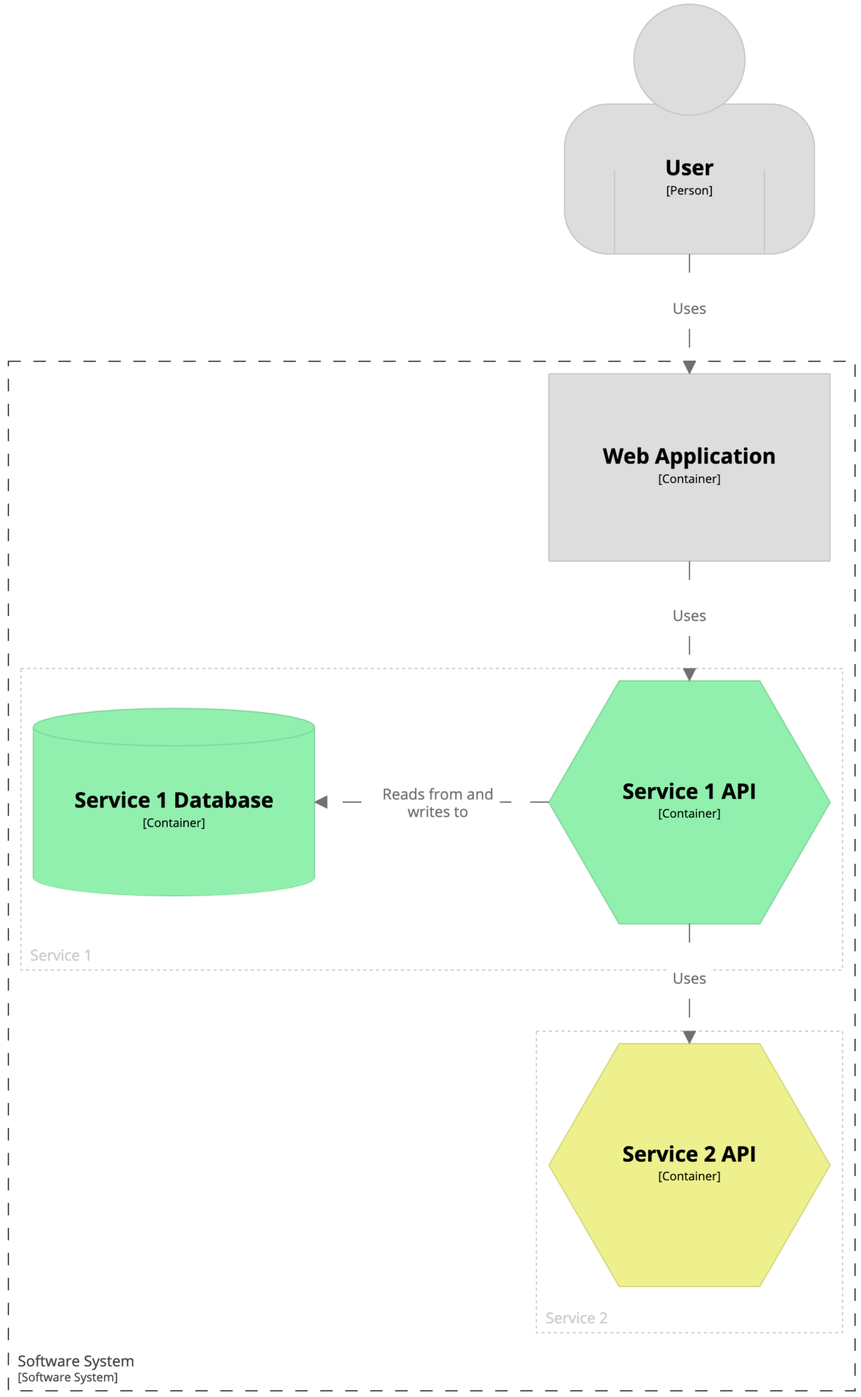
The container diagram for the Internet Banking System.
Tuesday, 24 March 2020, 18:29 Greenwich Mean Time

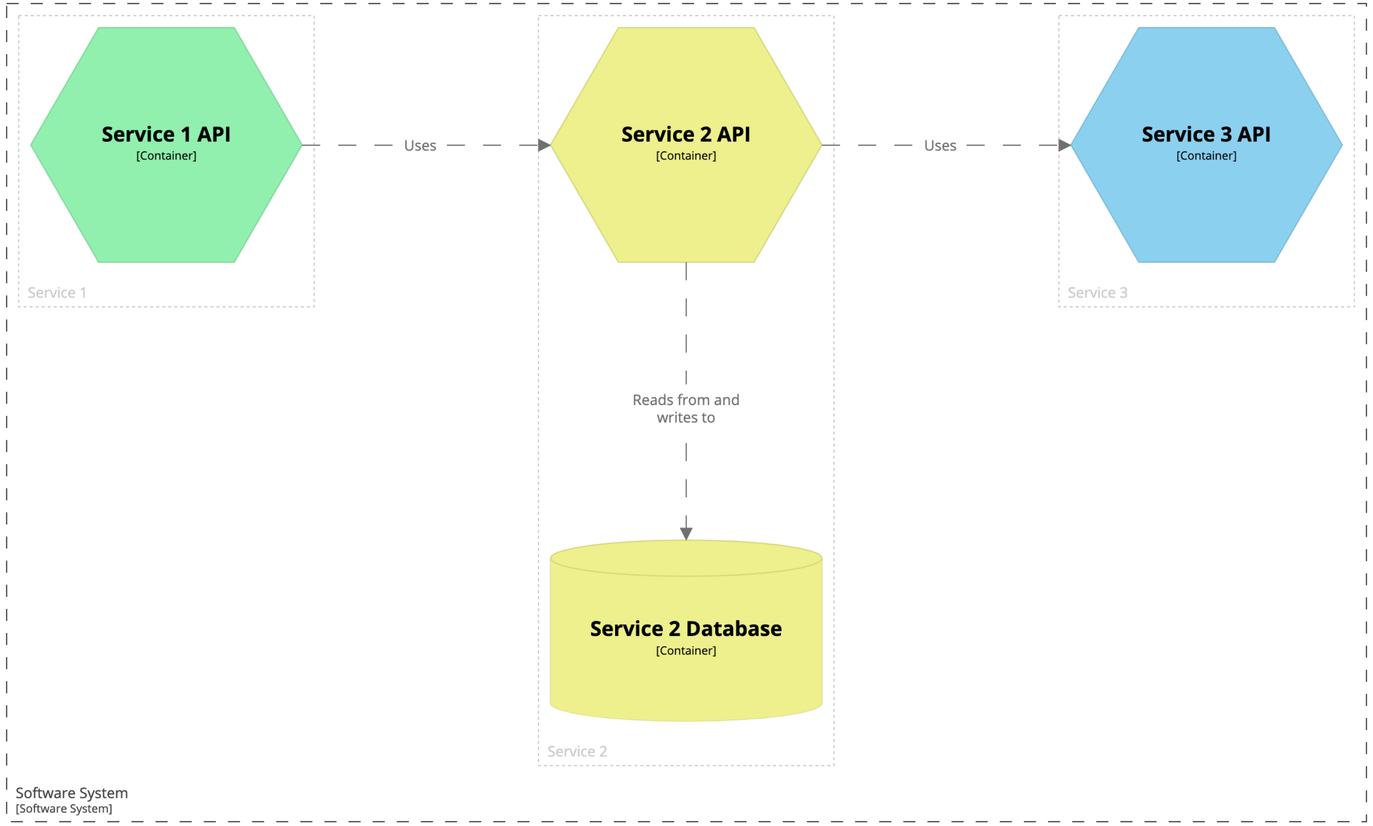


More advanced features

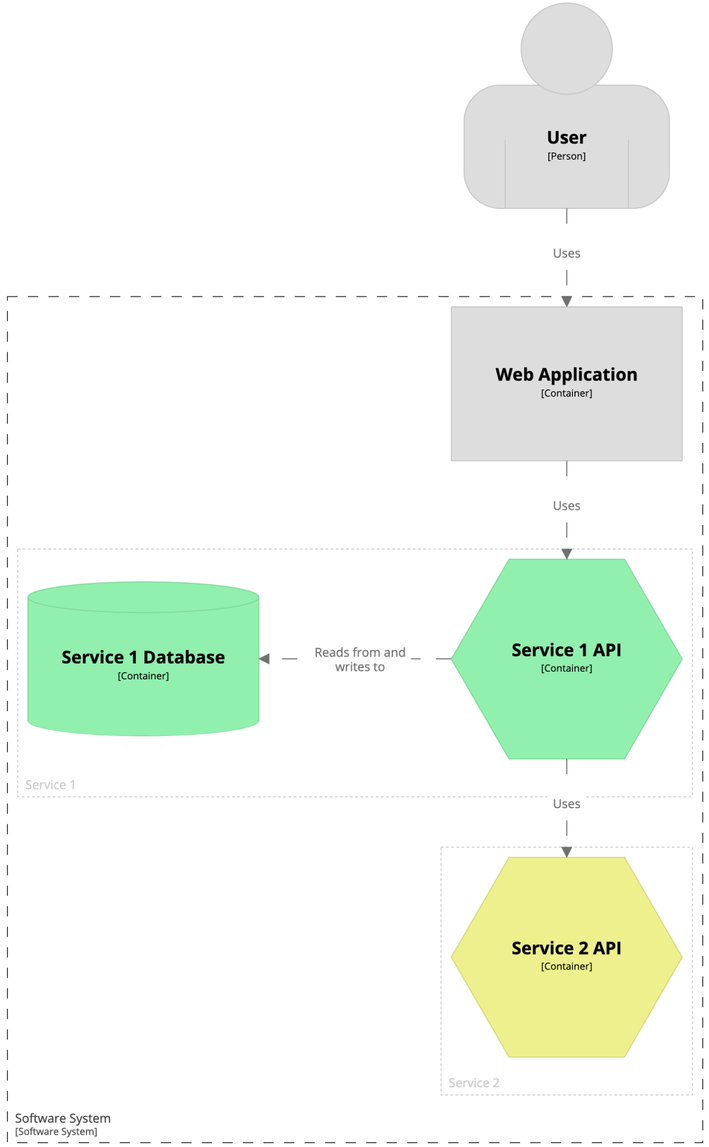
How do you diagram large and complex software systems?

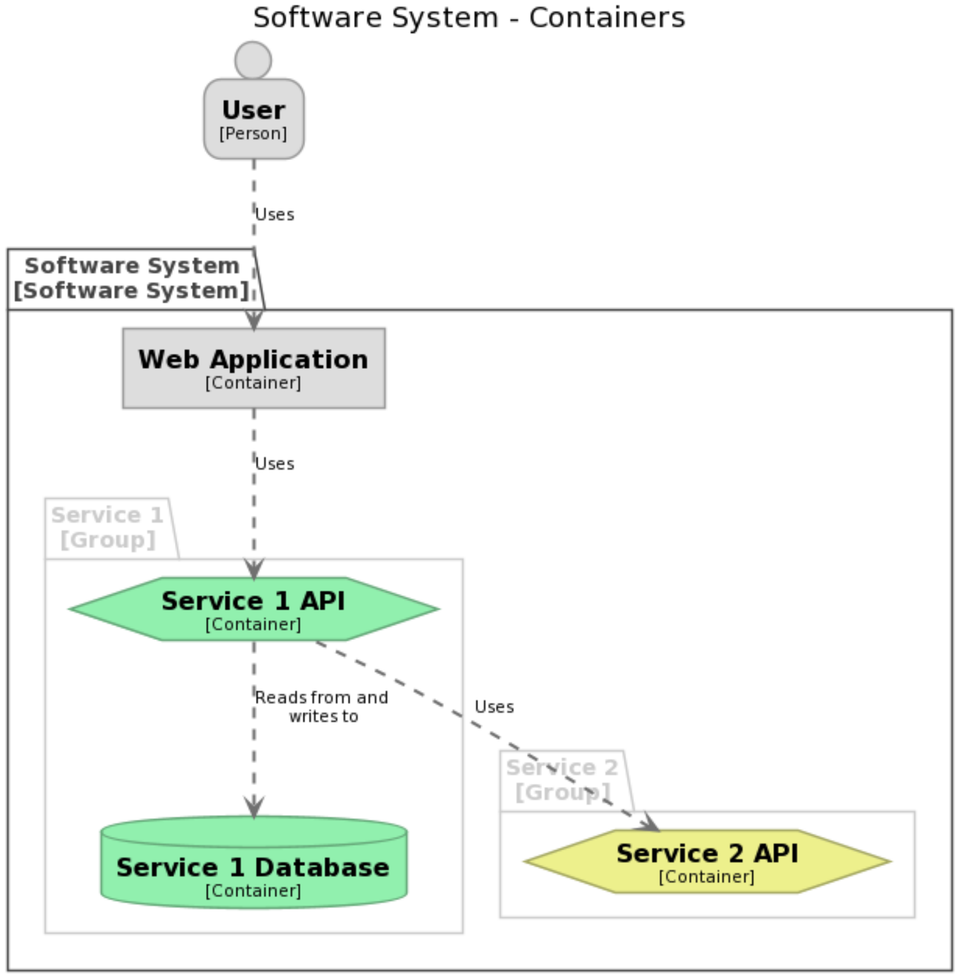

User
[Person]
Uses
Web Application
[Container]
Uses
Uses
Service 1 Database
[Container]
Reads from and writes to
Service 1 API
[Container]
Service 1
Service 3 API
[Container]
Reads from and writes to
Service 3 Database
[Container]
Service 3
Uses
Uses
Service 2 Database
[Container]
Reads from and writes to
Service 2 API
[Container]
Service 2
Service 4 API
[Container]
Reads from and writes to
Service 4 Database
[Container]
Service 4
Software System
[Software System]


User
[Person]
Uses
Web Application
[Container]
Uses
Service 1 Database
[Container]
Reads from and writes to
Service 1 API
[Container]
Service 1
Uses
Service 2 API
[Container]
Service 2
Software System
[Software System]

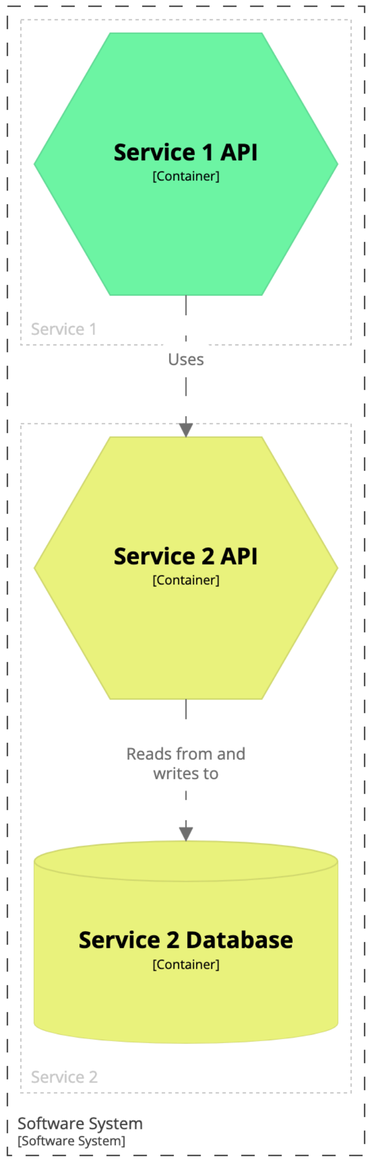

Service 1 API
[Container]
Uses
Service 2 API
[Container]
Uses
Service 3 API
[Container]
Service 1
Reads from and writes to
Service 2 Database
[Container]
Service 2
Service 3
Software System
[Software System]

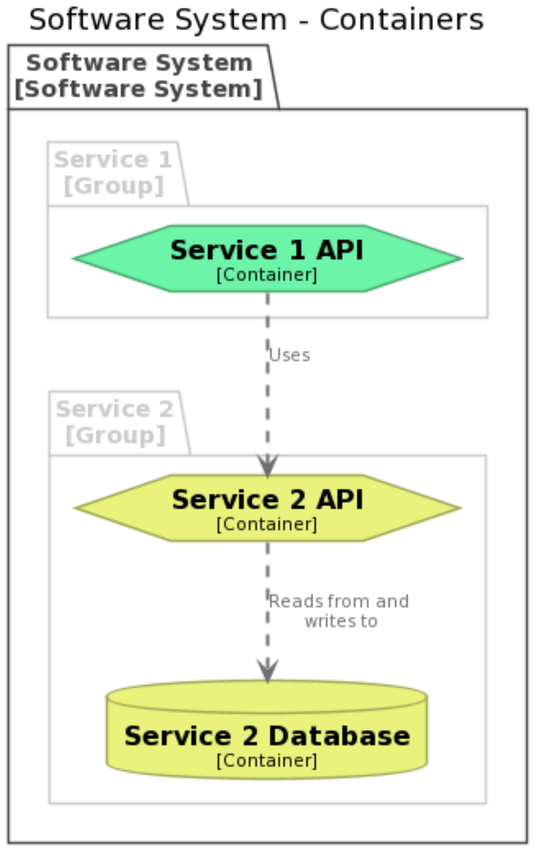
```
container softwareSystem {
    include user ->service1->
    autolayout
}
```

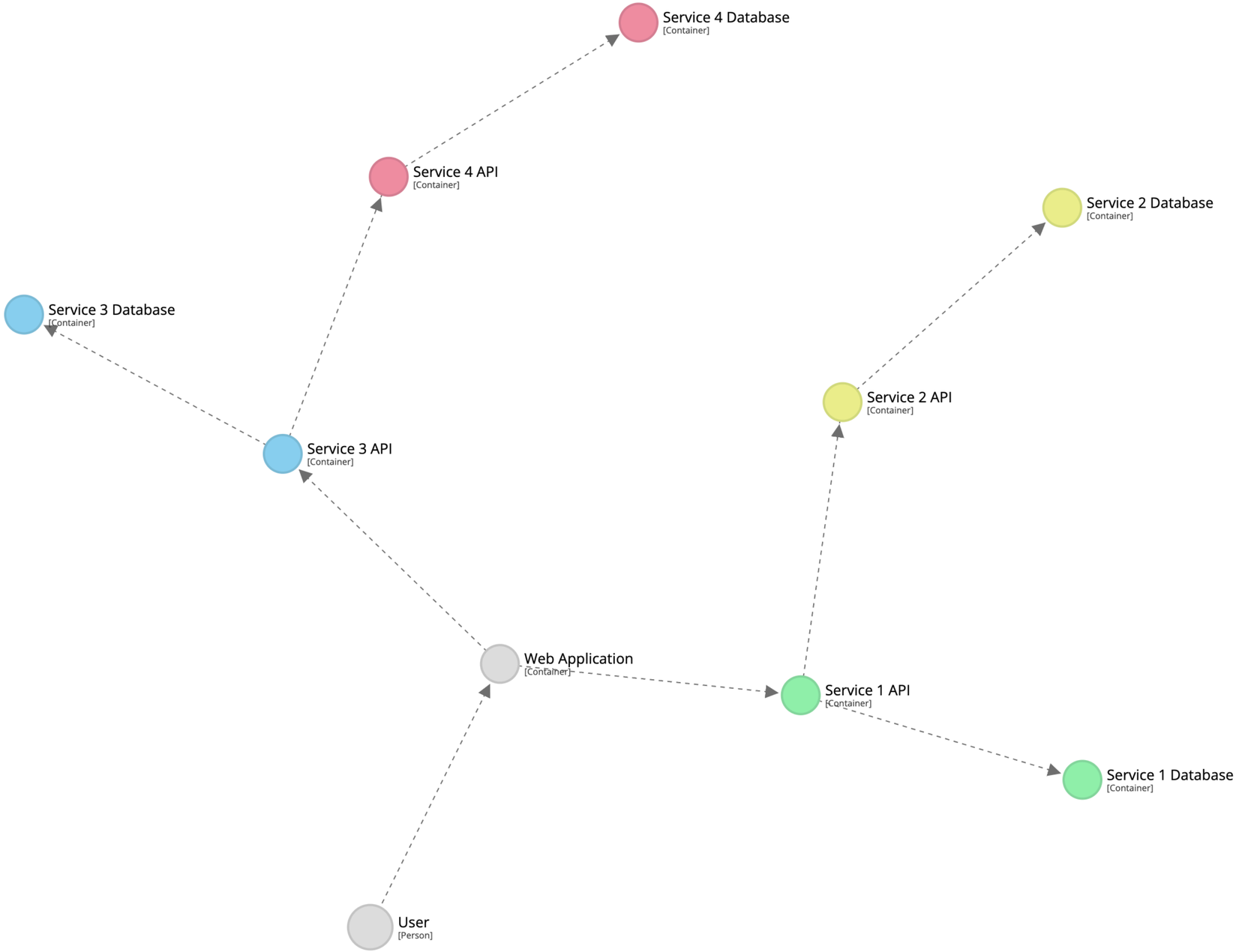


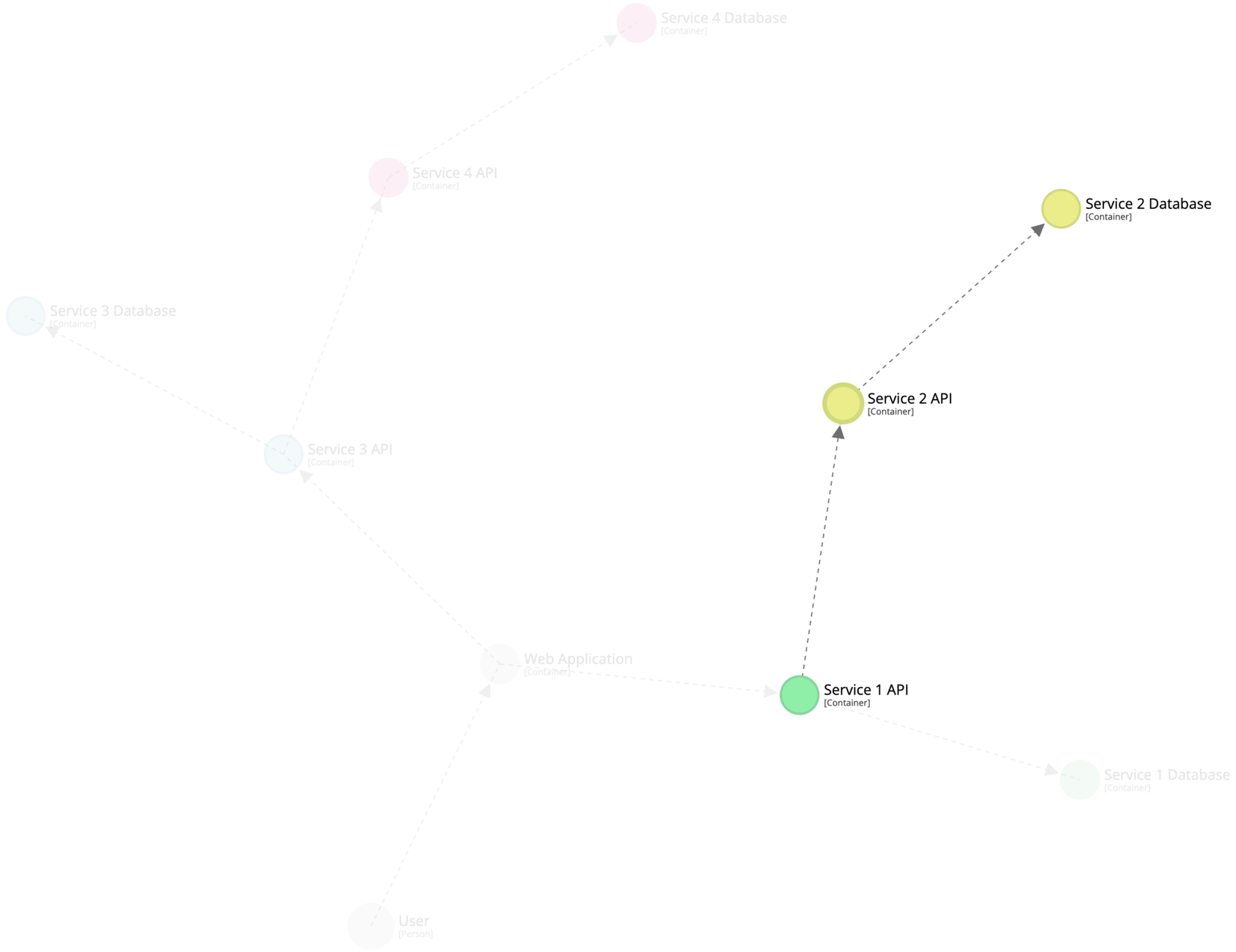


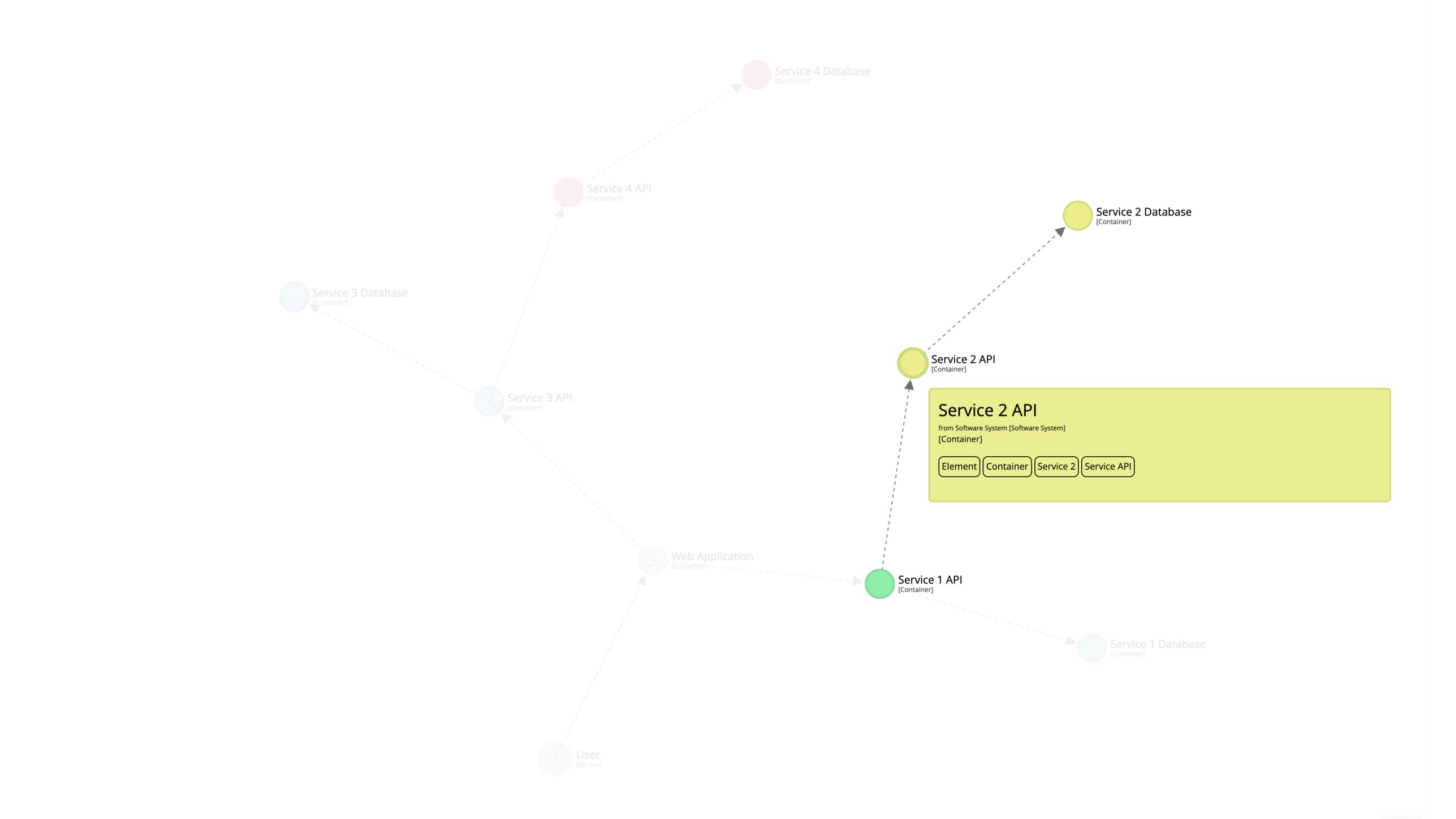
```
container softwareSystem {
    include ->service2->
    autolayout
}
```






Service 4 Database
[Container]
Service 4 API
[Container]
Service 2 Database
[Container]
Service 3 Database
[Container]
Service 2 API
[Container]
Service 3 API
[Container]
Web Application
[Container]
Service 1 API
[Container]
Service 1 Database
[Container]
User
[Person]


Service 4 Database
[Container]
Service 4 API
[Container]
Service 2 Database
[Container]
Service 3 Database
[Container]
Service 2 API
[Container]
Service 3 API
[Container]
Web Application
[Container]
Service 1 API
[Container]
Service 1 Database
[Container]
User
[Person]


Service 4 Database
[Container]
Service 4 API
[Container]
Service 2 Database
[Container]
Service 3 Database
[Container]
Service 2 API
[Container]
Service 2 API
from Software System [Software System]
[Container]
Element
Container
Service 2
Service API
Service 3 API
[Container]
Web Application
[Container]
Service 1 API
[Container]
Service 1 Database
[Container]
User
[Person]

Save changes

```yaml
resources:

  - id: "1"
    name: "User"
    subtitle: "[Person]"
    backgroundColor: "#dddddd"
    color: "#000000"

  - id: "2"
    name: "Software System"
    subtitle: "[Software System]"
    backgroundColor: "#dddddd"
    color: "#000000"

    children:
      - id: "10"
        name: "Service 3 API"
        subtitle: "[Container]"
        backgroundColor: "#8cd0f0"
        color: "#000000"

      - id: "11"
        name: "Service 3 Database"
        subtitle: "[Container]"
        backgroundColor: "#8cd0f0"
        color: "#000000"

      - id: "13"
        name: "Service 4 API"
        subtitle: "[Container]"
        backgroundColor: "#f08ca4"
        color: "#000000"

      - id: "14"
        name: "Service 4 Database"
        subtitle: "[Container]"
        backgroundColor: "#f08ca4"
        color: "#000000"

      - id: "3"
        name: "Web Application"
        subtitle: "[Container]"
        backgroundColor: "#dddddd"
        color: "#000000"

      - id: "4"
        name: "Service 1 API"
        subtitle: "[Container]"
        backgroundColor: "#91f0ae"
        color: "#000000"

      - id: "5"
        name: "Service 1 Database"
        subtitle: "[Container]"
        backgroundColor: "#91f0ae"
        color: "#000000"

      - id: "7"
        name: "Service 2 API"
```

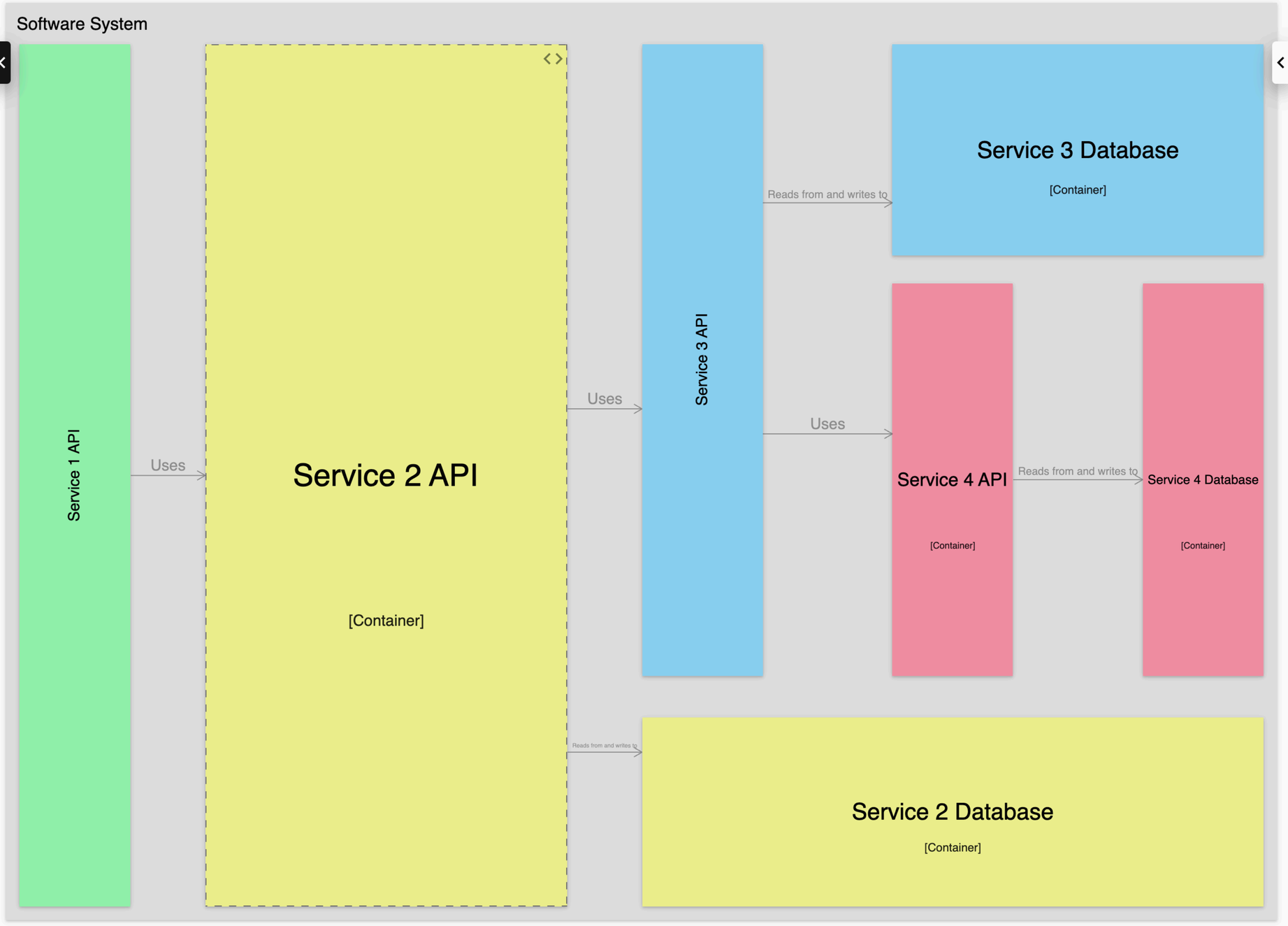
PROBLEMS 0 WARNINGS 0 SUGGESTIONS 2

Static Structure

Save changes

```
        name: "Service 2 Database"
        subtitle: "[Container]"
        backgroundColor: "#edf08c"
        color: "#000000"

      - id: "6"
        name: "Service 2 API"
        subtitle: "[Container]"
        backgroundColor: "#edf08c"
        color: "#000000"

      - id: "8"
        name: "Service 3 Database"
        subtitle: "[Container]"
        backgroundColor: "#8cd0f0"
        color: "#000000"

      - id: "9"
        name: "Service 3 API"
        subtitle: "[Container]"
        backgroundColor: "#8cd0f0"
        color: "#000000"

perspectives:
  - name: Static Structure
    relations:
      - from: "12"
        to: "11"
        label: "Reads from and writes to"

      - from: "3"
        to: "6"
        label: "Uses"

      - from: "3"
        to: "2"
        label: "Reads from and writes to"

      - from: "6"
        to: "9"
        label: "Uses"

      - from: "6"
        to: "5"
        label: "Reads from and writes to"

      - from: "9"
        to: "8"
        label: "Reads from and writes to"

      - from: "9"
        to: "12"
        label: "Uses"
```

PROBLEMS 0 WARNINGS 0 SUGGESTIONS 2



Static Structure

Enterprise-wide modelling?

Software systems and people

system-landscape.dsl

Software System 1

software-system-1.dsl
extends
system-landscape.dsl

Software System 2

software-system-2.dsl
extends
system-landscape.dsl

Software System 3

software-system-3.dsl
extends
system-landscape.dsl

model.dsl

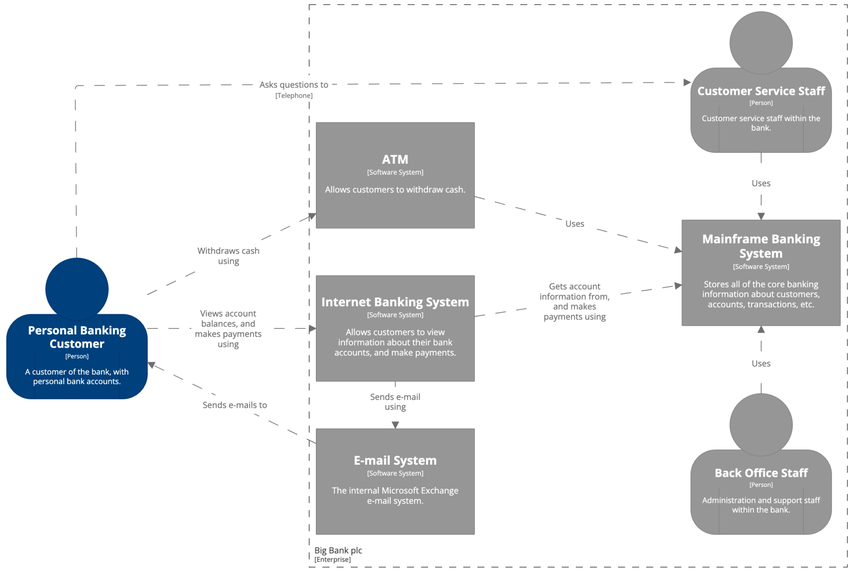
(people and software systems)

extends

extends

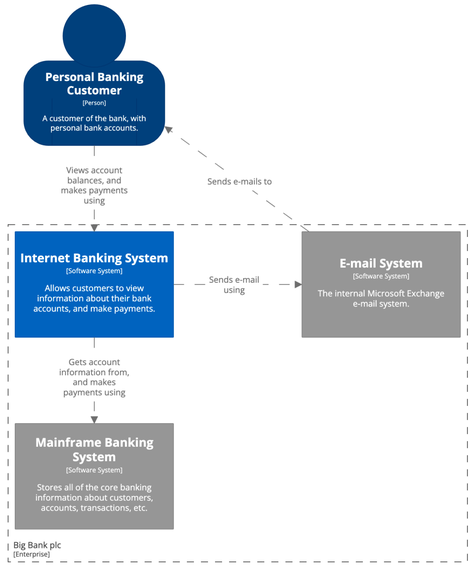
system-landscape.dsl

(system landscape diagrams)



internet-banking-system.dsl

(C4 model diagrams for the Internet Banking System)



Scripting support

(via JSR-223: Java Scripting API)

```
workspace {

    model {
        ...
    }

    !script groovy {
        workspace.views.createDefaultViews()
        workspace.views.views.each { it.disableAutomaticLayout() }
    }

}
```

Plugin support

(via Java)

Hybrid usage

(DSL and Java)

```
workspace {

    model {
        s = softwareSystem "Software System" {
            webapp = container "Web Application"
            database = container "Database" {
                webapp -> this "Reads from and writes to"
            }
        }
    }

    views {
        systemContext s {
            include *
            autoLayout
        }

        container s {
            include *
            autoLayout
        }
    }

}
```

```
StructurizrDslParser parser = new StructurizrDslParser();
parser.parse(new File("workspace.dsl"));

Workspace workspace = parser.getWorkspace();
Container webApplication = workspace.getModel()
        .getSoftwareSystemWithName("Software System”)
        .getContainerWithName(“Web Application");

// add components manually or via automatic extraction
...

// add a component view
ComponentView componentView = workspace.getViews()
        .createComponentView(webApplication, "Components", "Description");
componentView.addDefaultElements();
componentView.enableAutomaticLayout();
```

Custom tooling

Authoring tool

Create diagrams as code (Java, .NET, TypeScript, Python, PHP, etc) or text (DSL, YAML) via a number of different authoring tools.

Creates

Workspace

A workspace is the wrapper for a software architecture model and views, described using the C4 model and an open JSON data format.

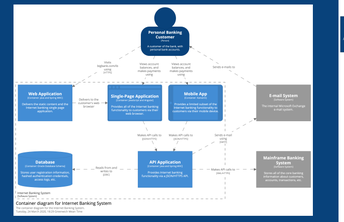
Renders

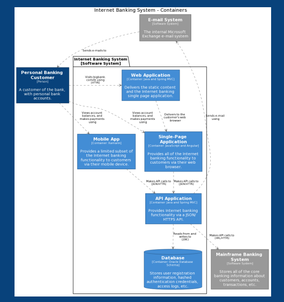
Rendering tool

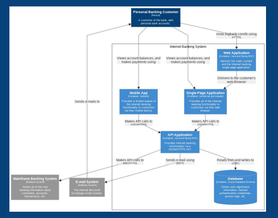
Render views using multiple diagramming tools and formats (Structurizr cloud service/on-premises installation, PlantUML, Mermaid, WebSequenceDiagrams, Ilograph, etc).

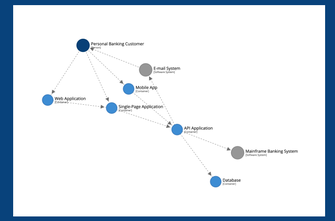
```java
private static final long WORKSPACE_ID = 25441;
private static final String API_KEY = "";
private static final String API_SECRET = "";

public static void main(String[] args) throws Exception {
    // all software architecture models belong to a workspace
    Workspace workspace = new Workspace("Getting Started", "This is a model of my software system.");
    Model model = workspace.getModel();

    // create a model to describe a user using a software system
    Person user = model.addPerson("User", "A user of my software system.");
    SoftwareSystem softwareSystem = model.addSoftwareSystem("Software System", "My software system.");
    user.uses(softwareSystem, "Uses");

    // create a system context diagram showing people and software systems
    ViewSet views = workspace.getViews();
    SystemContextView contextView = views.createSystemContextView(softwareSystem, "SystemContext", "An example of
    contextView.addAllSoftwareSystems();
    contextView.addAllPeople();

    // add some styling to the diagram elements
    Styles styles = views.getConfiguration().getStyles();
    styles.addElementStyle(Tags.SOFTWARE_SYSTEM).background("#1168bd").color("#ffffff");
    styles.addElementStyle(Tags.PERSON).background("#08427b").color("#ffffff").shape(Shape.Person);

    // upload to structurizr.com (you'll need your own workspace ID, API key and API secret)
    StructurizrClient structurizrClient = new StructurizrClient(API_KEY, API_SECRET);
    structurizrClient.putWorkspace(WORKSPACE_ID, workspace);
}
```

```
workspace "Getting Started" "This is a model of my software system." {

    model {
        user = person "User" "A user of my software system."
        softwareSystem = softwareSystem "Software System" "My software system."

        user -> softwareSystem "Uses"
    }

    views {
        systemContext softwareSystem "SystemContext" "An example of a System Context diagram." {
            include *
        }

        styles {
            element "Software System" {
                background #1168bd
                color #ffffff
            }
            element "Person" {
                shape person
                background #08427b
                color #ffffff
            }
        }
    }

}
```









Consumes

Custom tool

Your own tooling to parse the model and views; for integration into other rendering tools, dashboards, service catalogs, etc.

Usage scenarios

Hand-crafted models

“Diagrams as data”

Static diagrams

(e.g. PNG/SVG)

Interactive diagrams

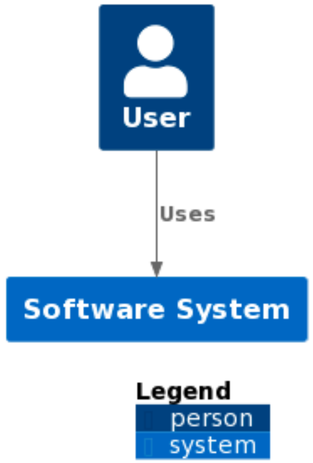
(e.g. browser-based)

Closing thoughts

“Diagrams as code” is easy to author, diff, version control, collaborate on, integrate into CI/CD, etc

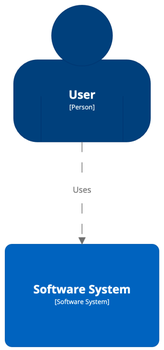
Diagrams as code 1.0

```
@startuml
!include https://raw.githubusercontent.com/plantuml-stdlib/C4-PlantUML/master/C4.puml
!include https://raw.githubusercontent.com/plantuml-stdlib/C4-PlantUML/master/C4_Context.puml

Person(User, "User")
System(SoftwareSystem, "Software System")

Rel_D(User, SoftwareSystem, "Uses")

SHOW_LEGEND()
@enduml
```

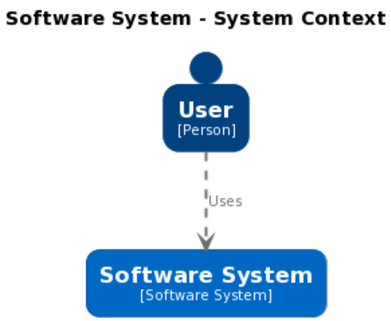


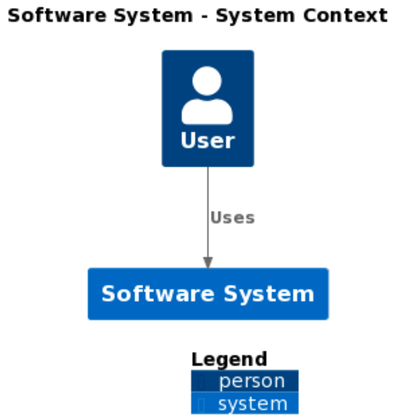
Diagrams as code 2.0

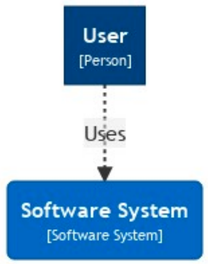
```
workspace {

    model {
        user = person "User"
        softwareSystem = softwareSystem "Software System"

        user -> softwareSystem "Uses"
    }

    views {
        theme default
    }

}
```

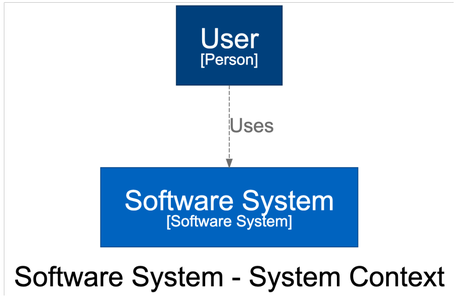


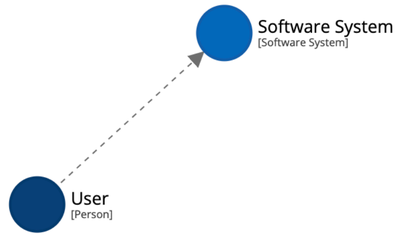
[System Context] Software System
Friday, 4 March 2022, 09:27 Greenwich Mean Time

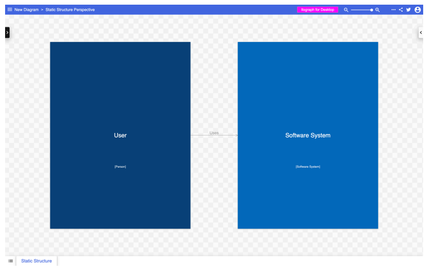












“Diagrams as code 2.0”
makes this model based,
separating content from presentation

Think about diagrams as being “disposable” artefacts

Developers
vs
non-developers?

Store your diagrams and docs
in version control,
next to your source code

“Publish” the diagrams and documentation if necessary

Up front design

vs

long-lived documentation?

README.md

Structurizr DSL cookbook

Creating software architecture diagrams from a textual definition is becoming more popular, but it's easy to introduce inconsistencies into your diagrams if you don't keep the multiple source files in sync. This cookbook is a guide to the Structurizr DSL, an open source tool for creating diagrams as code from a single consistent model.

Table of contents

https://github.com/structurizr/dsl/tree/master/docs/cookbook

The Structurizr DSL (as mentioned on the ThoughtWorks Tech Radar - Techniques - Diagrams as code) allows you to create multiple diagrams based upon the C4 model, in multiple output formats, from a single DSL source file. **Some features (`!docs`, `!adrs`, `!script`, etc) are unavailable on this demo page** - see Help - DSL for details.

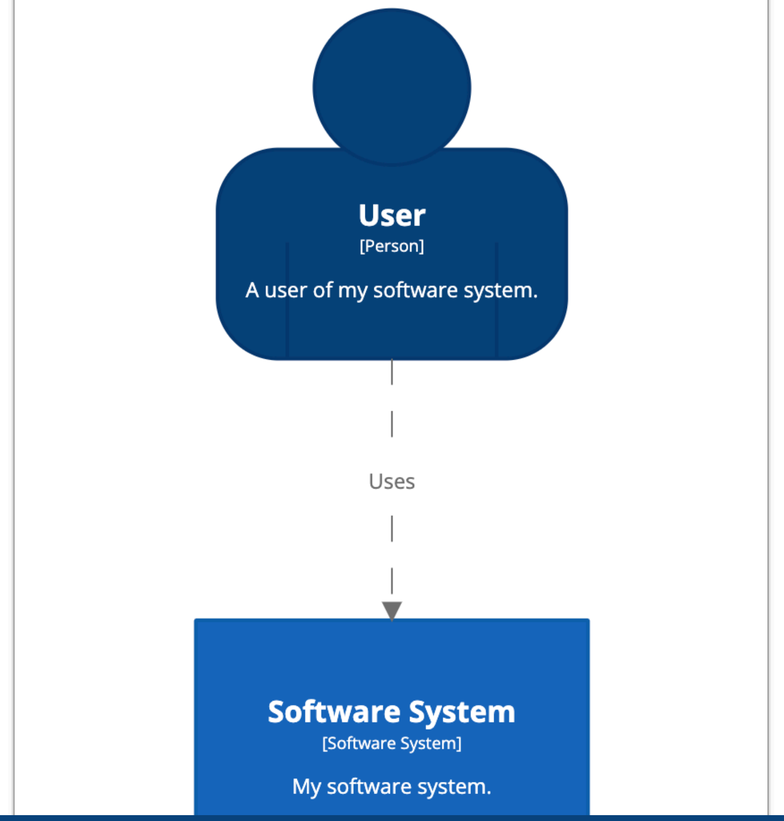
DSL language reference | Upload | </> | Render

```
1  workspace "Getting Started" "This is a model of my software system." {
2
3      model {
4          user = person "User" "A user of my software system."
5          softwareSystem = softwareSystem "Software System" "My software system."
6
7          user -> softwareSystem "Uses"
8      }
9
10     views {
11         systemContext softwareSystem "SystemContext" "An example of a System Context di
12             include *
13             autoLayout
14         }
15
16         styles {
17             element "Software System" {
18                 background #1168bd
19                 color #ffffff
20             }
21             element "Person" {
22                 shape person
23                 background #08427b
24                 color #ffffff
25             }
26         }
27     }
28
29 }
```

Structurizr Diagram | Structurizr Graph | Export PlantUML | Export C4-PlantUML | Export Mermaid | Export DOT | Export WebSequenceDiagrams | Export Ilograph

[System Context] Software System (#SystemContext)

(diagram not editable; automatic layout enabled)



https://structurizr.com/dsl

Thank you!

Simon Brown
@simonbrown