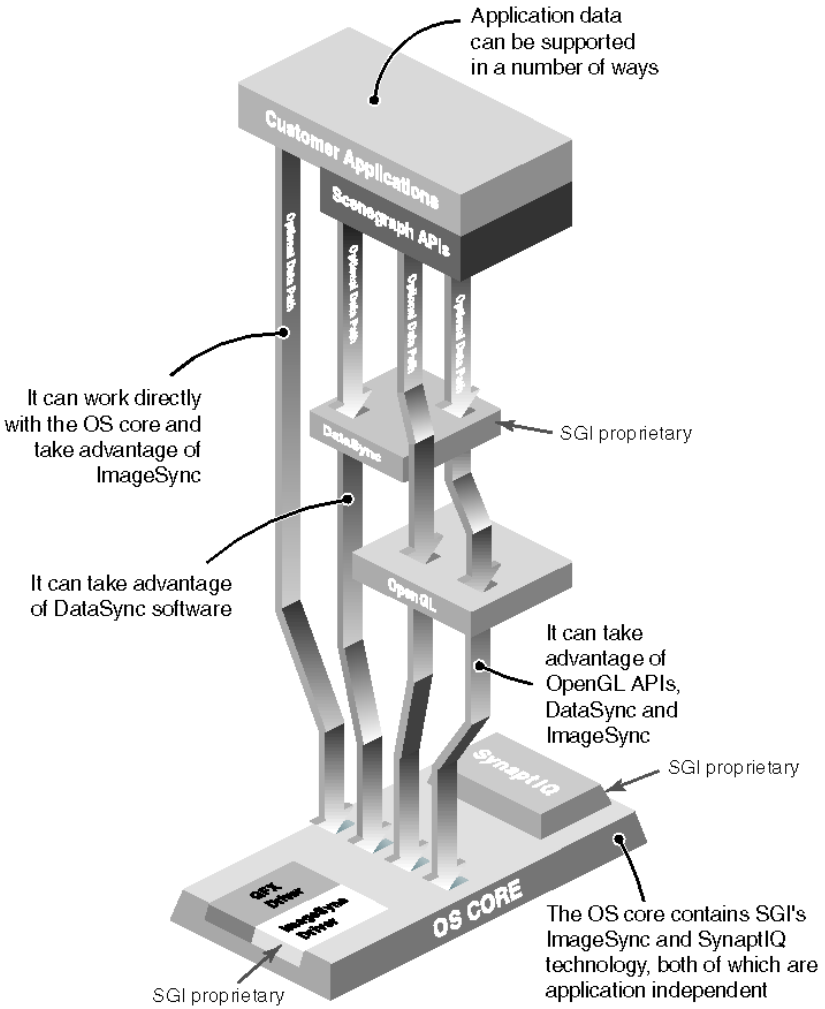
Arquitectura de Software

¿De qué estamos hablando ?

- estructura: descomposición en componentes y sus interacciones
- grandes decisiones de diseño basadas en requerimientos (incluyendo los no funcionales)



Las decisiones de arquitectura tendrán un impacto en el software

- performance - minimizar el número de comunicaciones
- security - organización en capas
- availability - redundancia controlada
- maintainability - componentes autocontenidas fácilmente intercambiables

Aspectos que influencian la arquitectura

- requerimientos funcionales
- atributos de calidad (performance, seguridad, etc)
- restricciones
- principios (consistencia y claridad del código)

Atributos de Calidad

- performance (tiempo de respuesta, latencia, etc)
- escalabilidad
- disponibilidad
- seguridad
- continuidad operacional (recuperación)
- acccesibilidad (usuarios con necesidades)

Algunas Restricciones

- Tiempo y presupuesto (las obvias)
- Tecnológicas
  - lista de tecnologías aprobadas
  - sistemas existentes e interoperabilidad
  - plataforma destino
  - open source (yes/no)
  - madurez de tecnologías a incluir
  - relaciones con los proveedores
  - fallas anteriores
- Relativas a personas
  - tamaño del equipo
  - habilidades del equipo
  - se pueden agregar especialistas
  - la mantención la hará el mismo equipo ?

Algunos Principios

- De Desarrollo
  - estandares de codificación
  - full unit testing
  - revisión de código
- Arquitectónicos
  - separación de lógica de negocios
  - componentes sin estado
  - consistencia eventual
  - no usamos procedimientos almacenados
  - gestión del sistema
  - auditoría flexibilidad
  - aspectos legales y regulatorios
  - localización e internacionalización

Arquitectura vs Diseño

- La separación no es tan nítida pero hay aspectos que son claramente de arquitectura
  - la "forma" del sistema: client-server, web-based, native mobile client, etc
  - estructura del software (componentes, capas)
  - tecnologías (lenguaje, plataforma de despliegue)
  - frameworks
  - enfoque de logro de performance, escalabilidad, etc

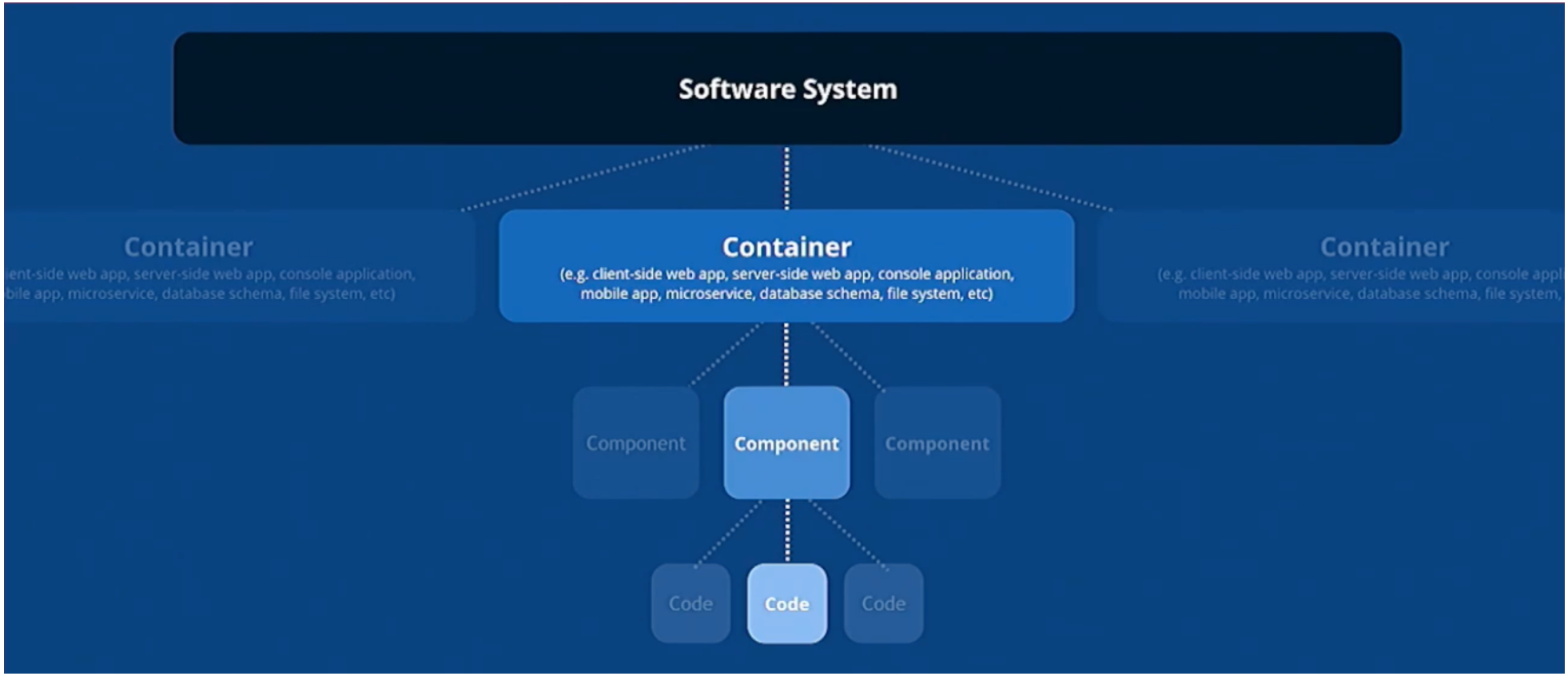
Arquitectura y Agilidad

- Desarrollo ágil NO implica olvidarse de la arquitectura
- Una buena arquitectura habilita la agilidad
  - por ejemplo arquitectura de microservicios (mas adelante) facilita desarrollo incremental
  - separación clara de componentes facilita los tests
  - se puede agregar funcionalidades con mayor facilidad sobre arquitectura sólida

Cómo describir la Arquitectura

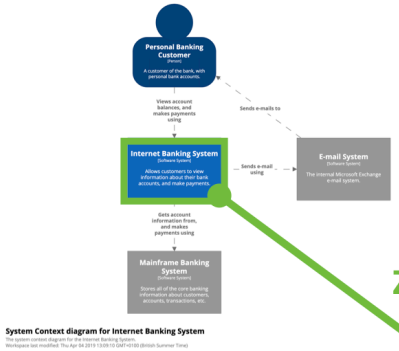
- Son fundamentales los diagramas
- Puede usarse UML (despliegue, componentes)
- Generalmente se usan diagramas más simples
- Descripción en 4 niveles: Contexto, Contenedores, Componentes, Code (Modelo C4)

El Modelo C4 (Simon Brown, 2011) (c4model.com)

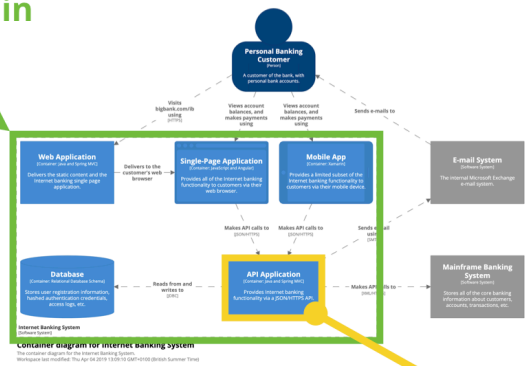


Son mas importantes las abstracciones que la notación específica
c4model.com

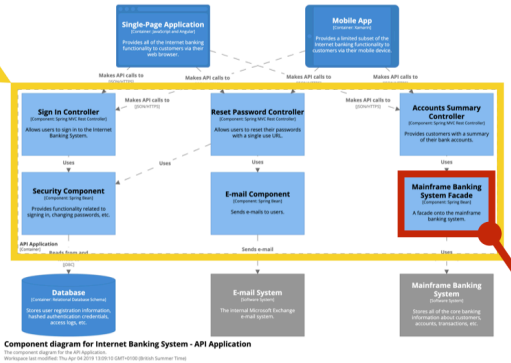
El Modelo C4



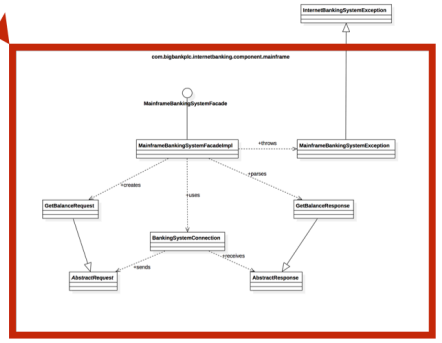
Zoom in



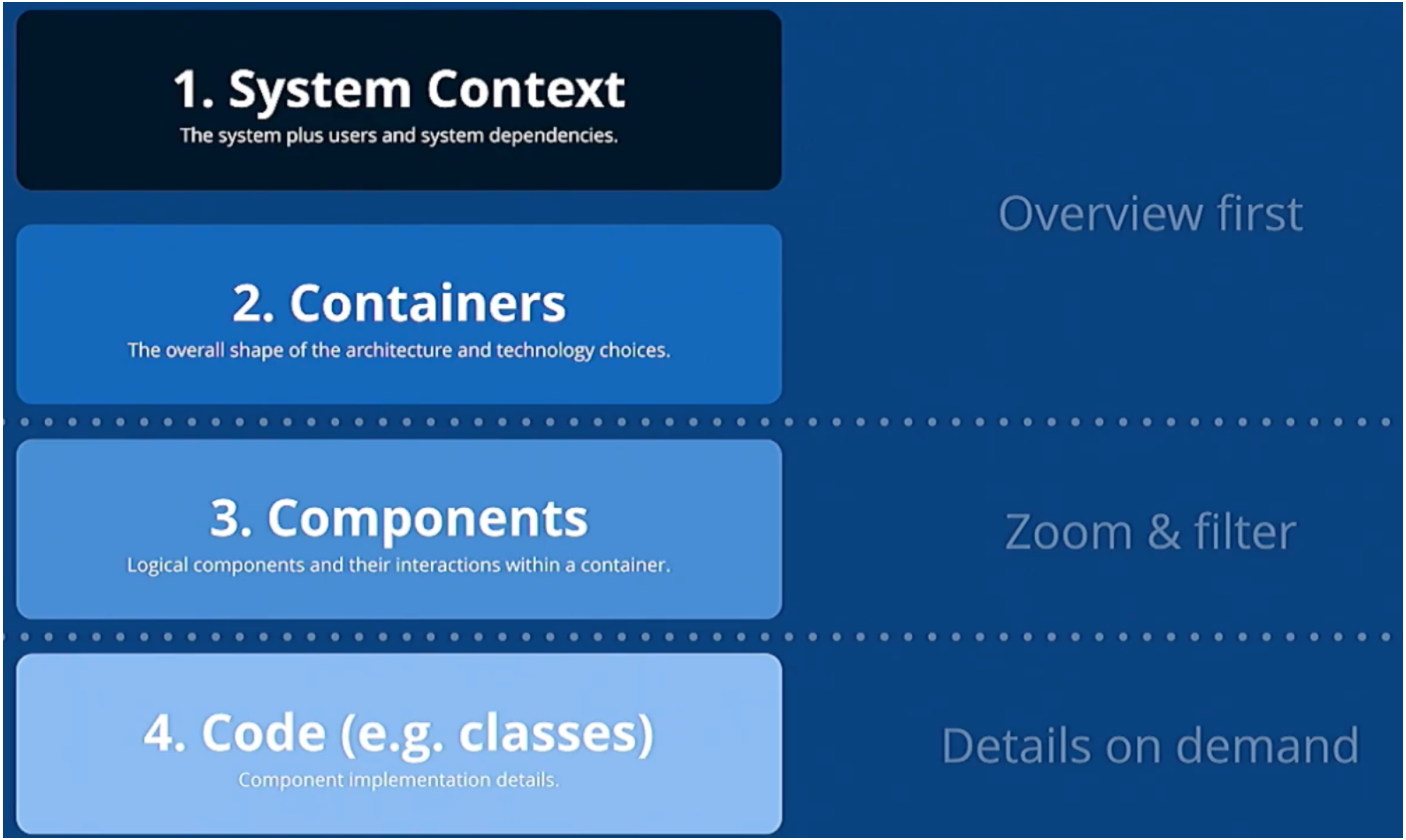
Zoom in



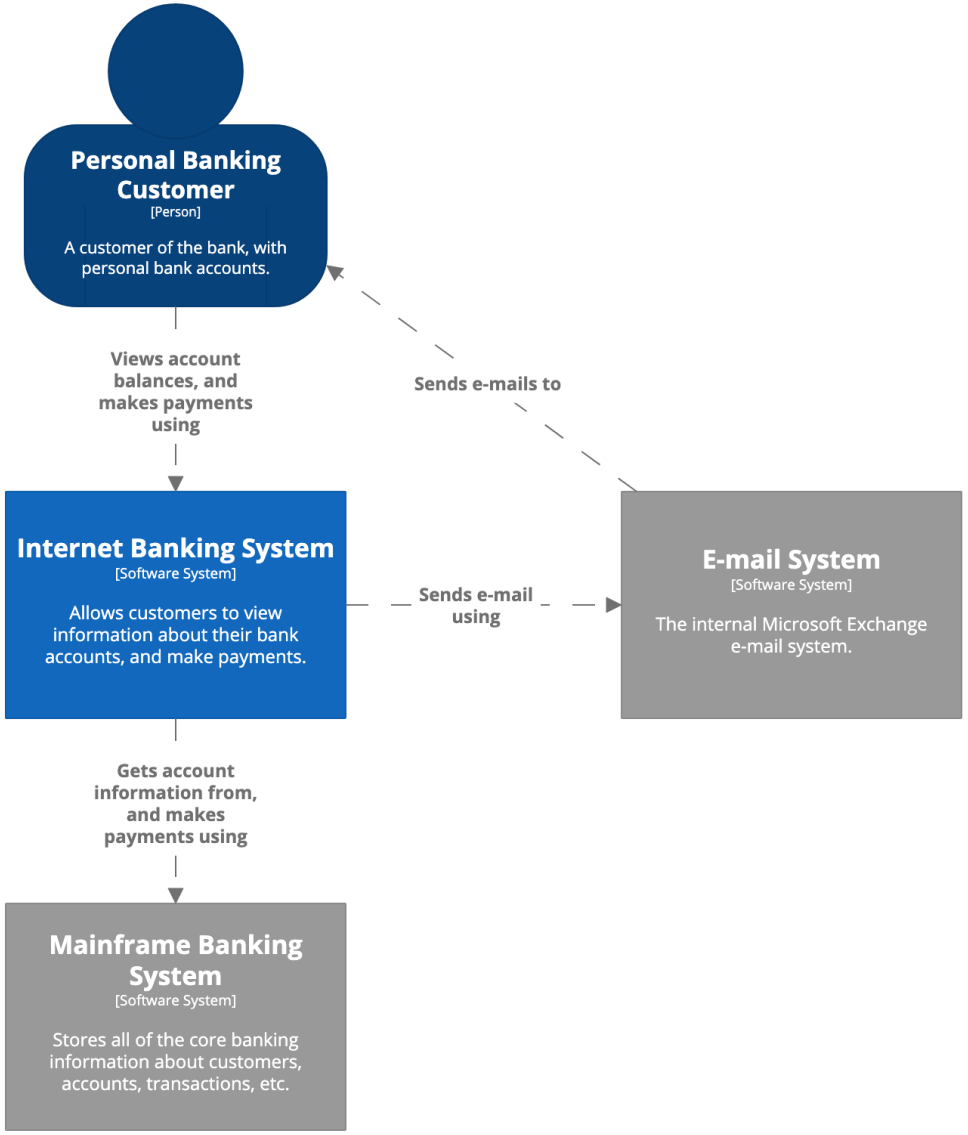
Zoom in



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


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. Code (e.g. classes)
Component implementation details.
Overview first
Zoom & filter
Details on demand

Context Diagram

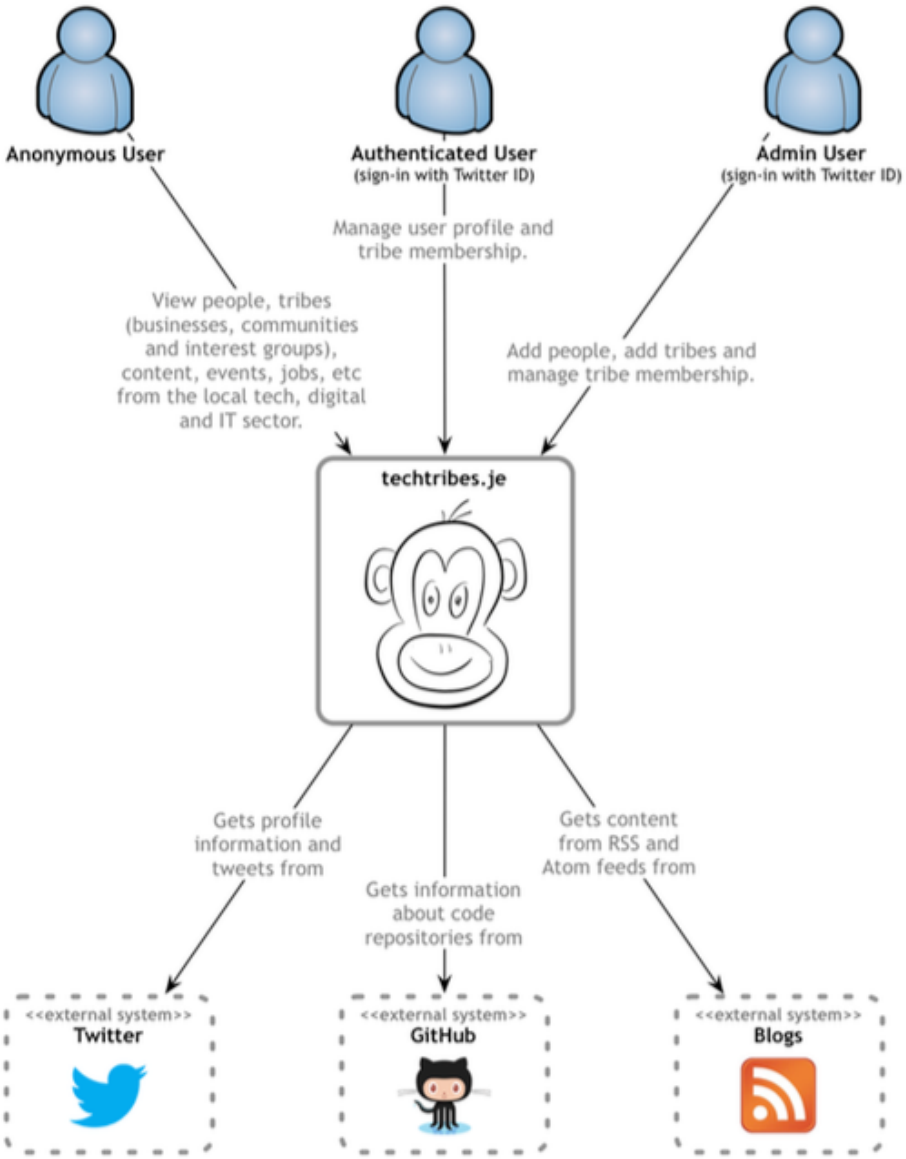


System Context diagram for Internet Banking System

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

Contexto

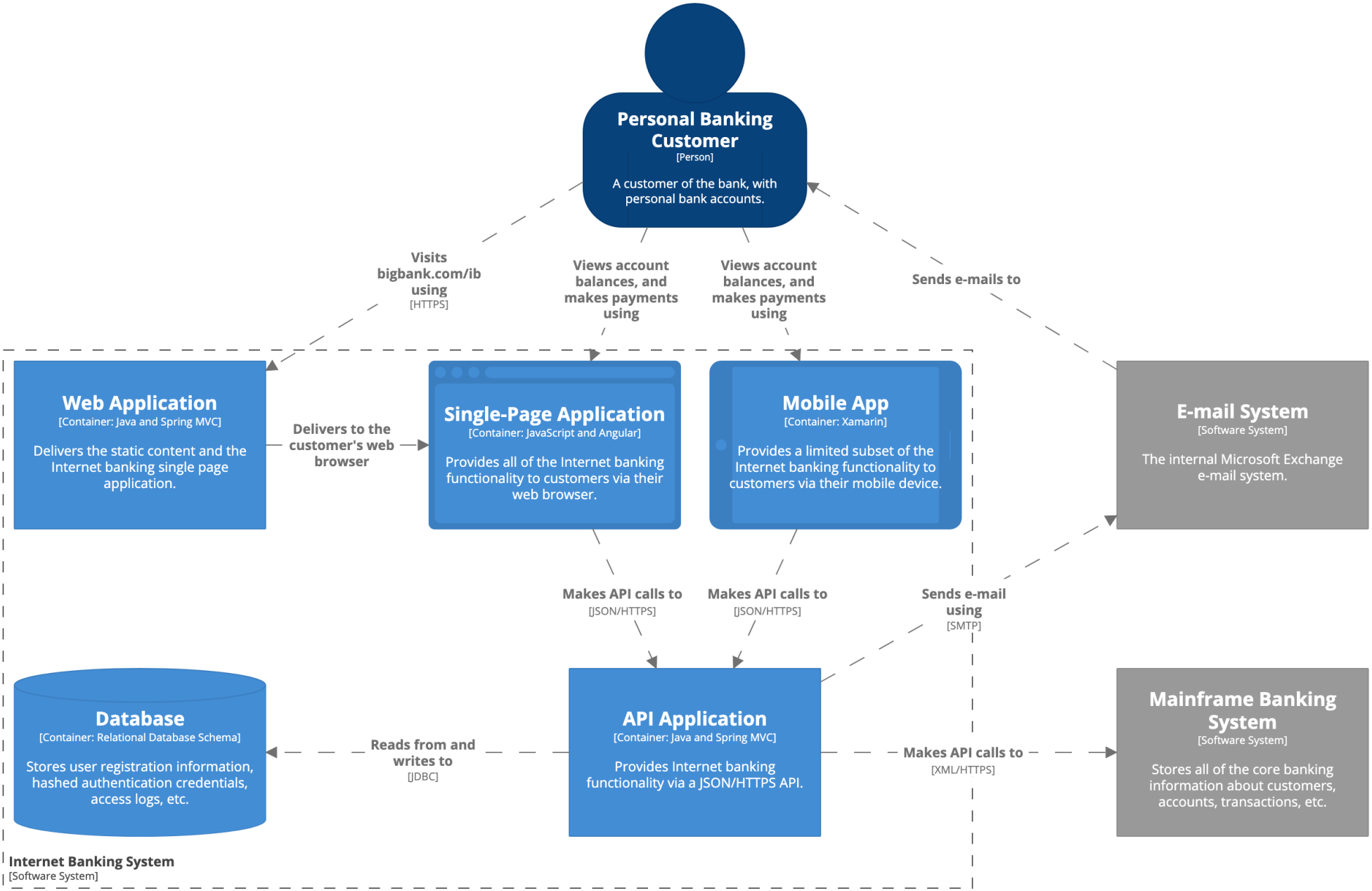
- Quien o quienes lo usarán
- De que se trata lo que estamos construyendo
- Como encaja en la infraestructura existente



Container Diagram

- Forma general del sistema
- decisiones tecnológicas de alto nivel
- distribución de responsabilidades
- comunicación entre containers
- donde se debe desarrollar qué código

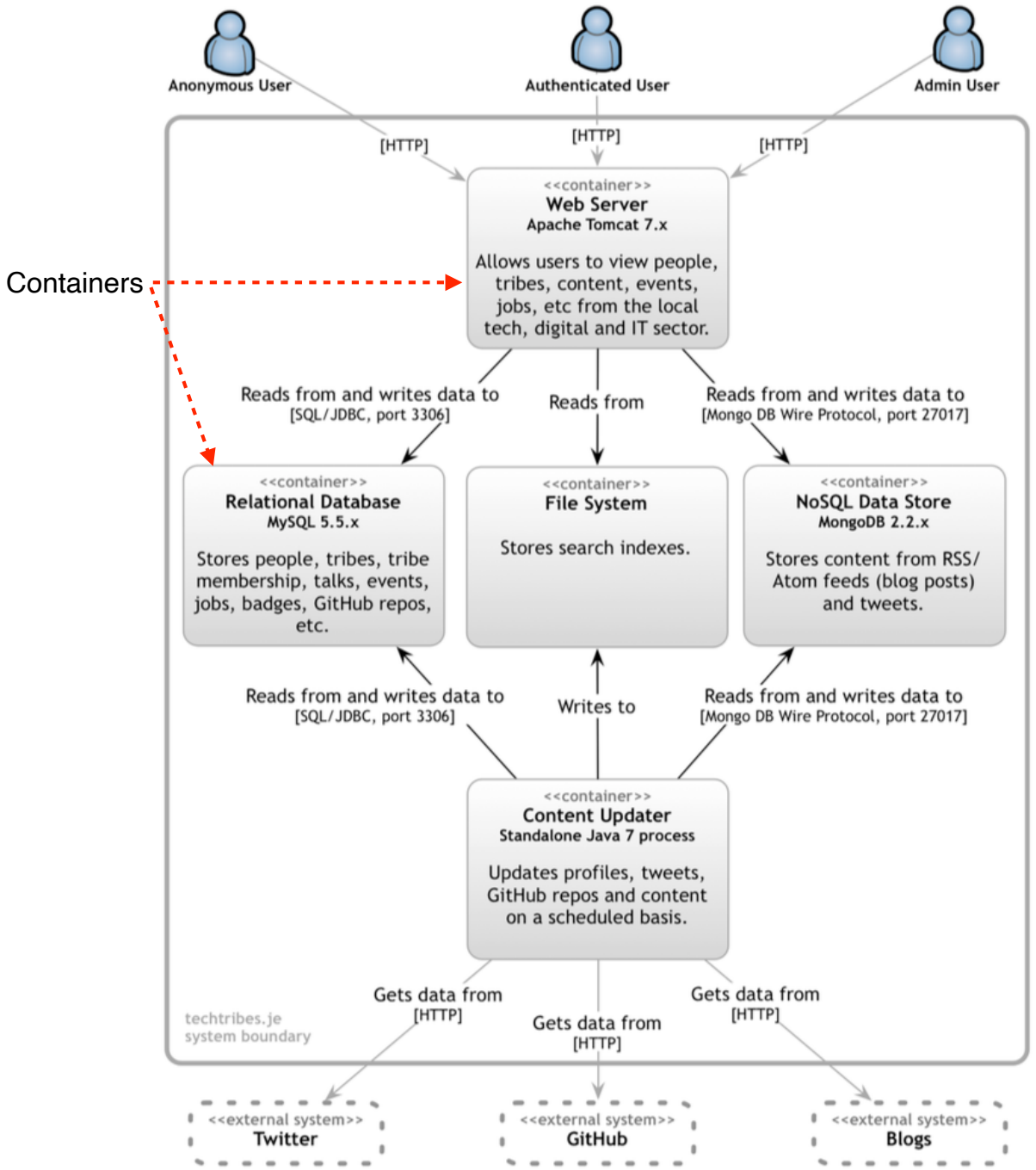
Internet Banking System



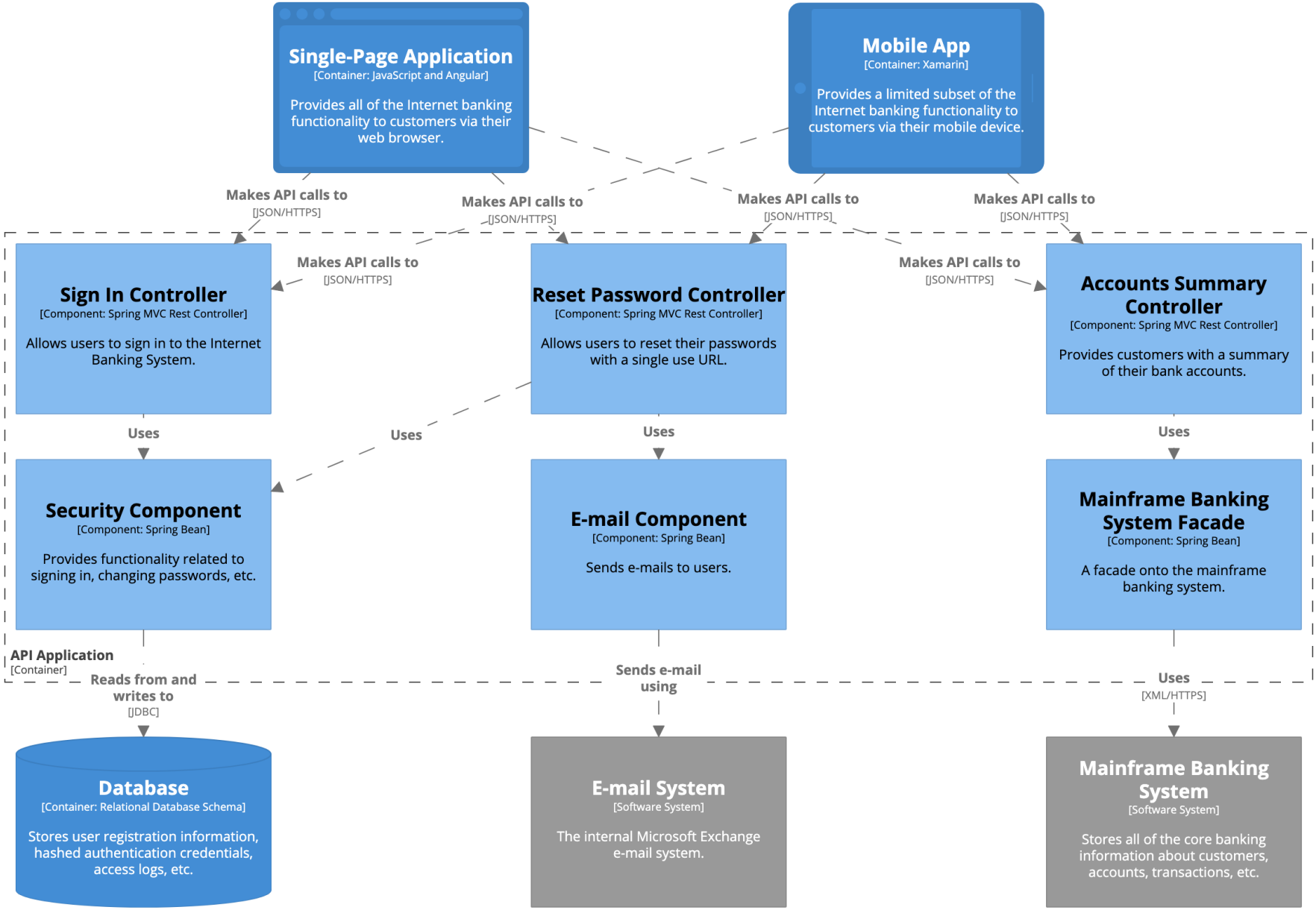
Container diagram for Internet Banking System

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

TechTribes

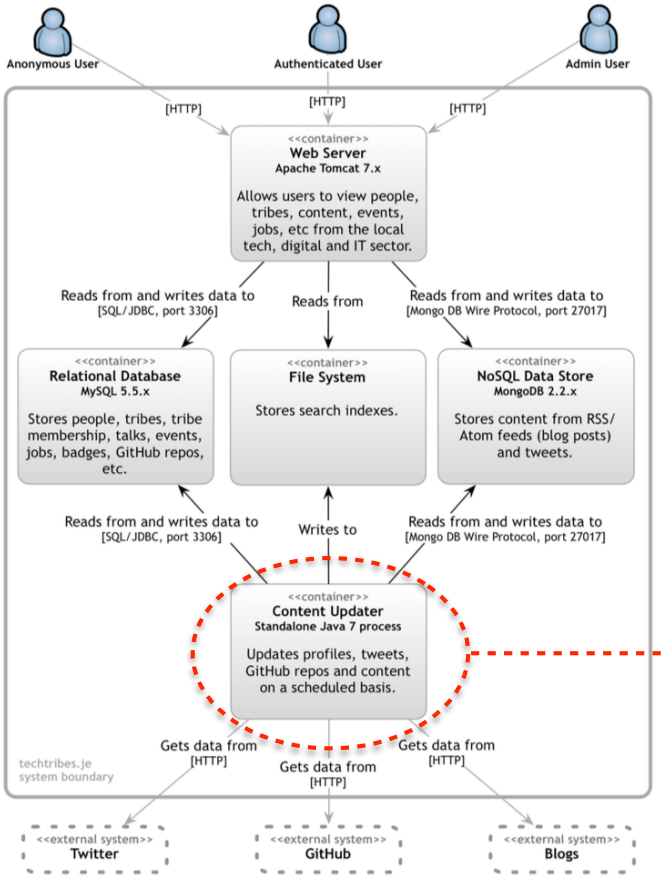

Containers
Anonymous User
Authenticated User
Admin User
[HTTP]
[HTTP]
[HTTP]
<<container>>
Web Server
Apache Tomcat 7.x
Allows users to view people, tribes, content, events, jobs, etc from the local tech, digital and IT sector.
Reads from and writes data to
[SQL/JDBC, port 3306]
Reads from
Reads from and writes data to
[Mongo DB Wire Protocol, port 27017]
<<container>>
Relational Database
MySQL 5.5.x
Stores people, tribes, tribe membership, talks, events, jobs, badges, GitHub repos, etc.
<<container>>
File System
Stores search indexes.
<<container>>
NoSQL Data Store
MongoDB 2.2.x
Stores content from RSS/ Atom feeds (blog posts) and tweets.
Reads from and writes data to
[SQL/JDBC, port 3306]
Writes to
Reads from and writes data to
[Mongo DB Wire Protocol, port 27017]
<<container>>
Content Updater
Standalone Java 7 process
Updates profiles, tweets, GitHub repos and content on a scheduled basis.
Gets data from
[HTTP]
Gets data from
[HTTP]
Gets data from
[HTTP]
techtribes.je
system boundary
<<external system>>
Twitter
<<external system>>
GitHub
<<external system>>
Blogs

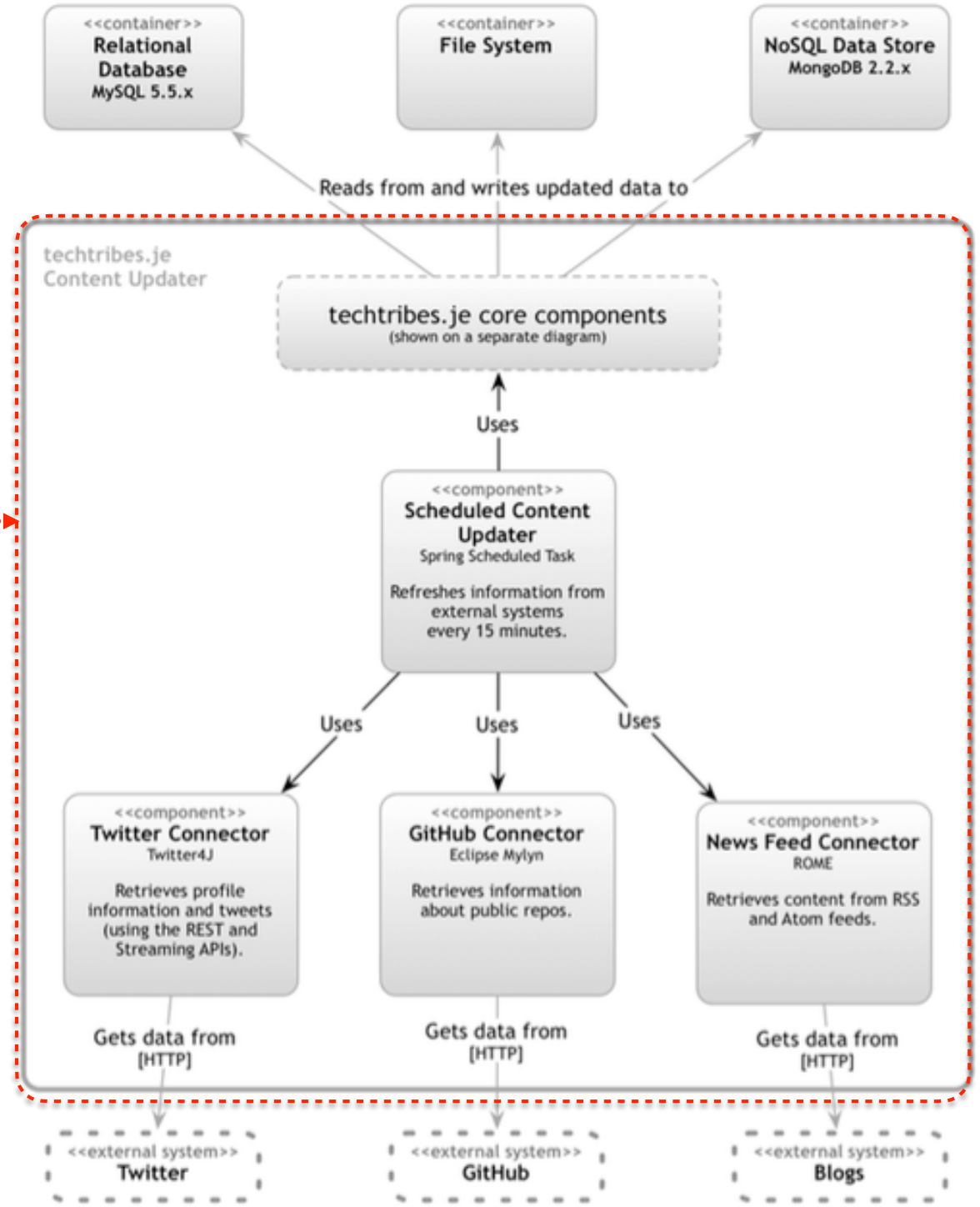
Components Diagram

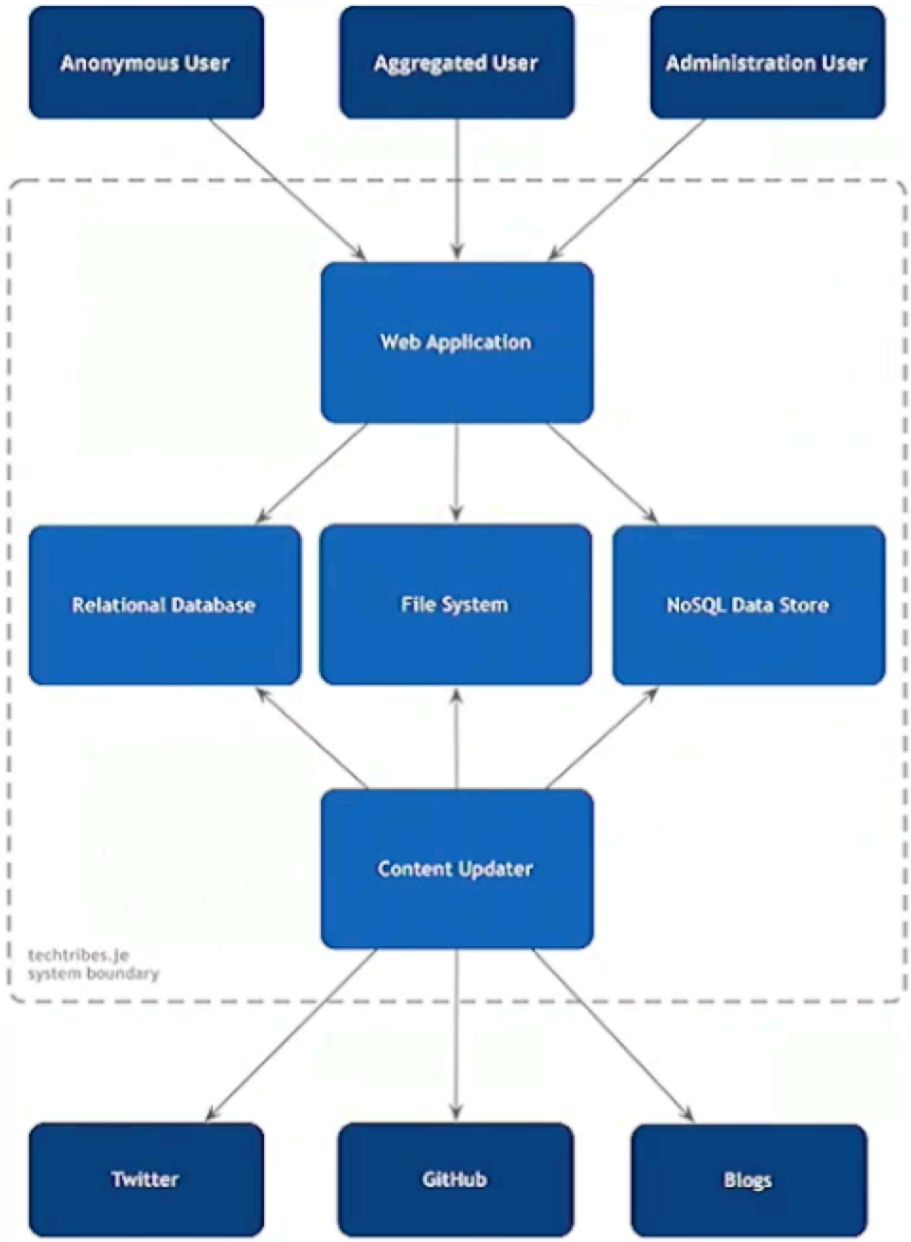


Component diagram for Internet Banking System - API Application

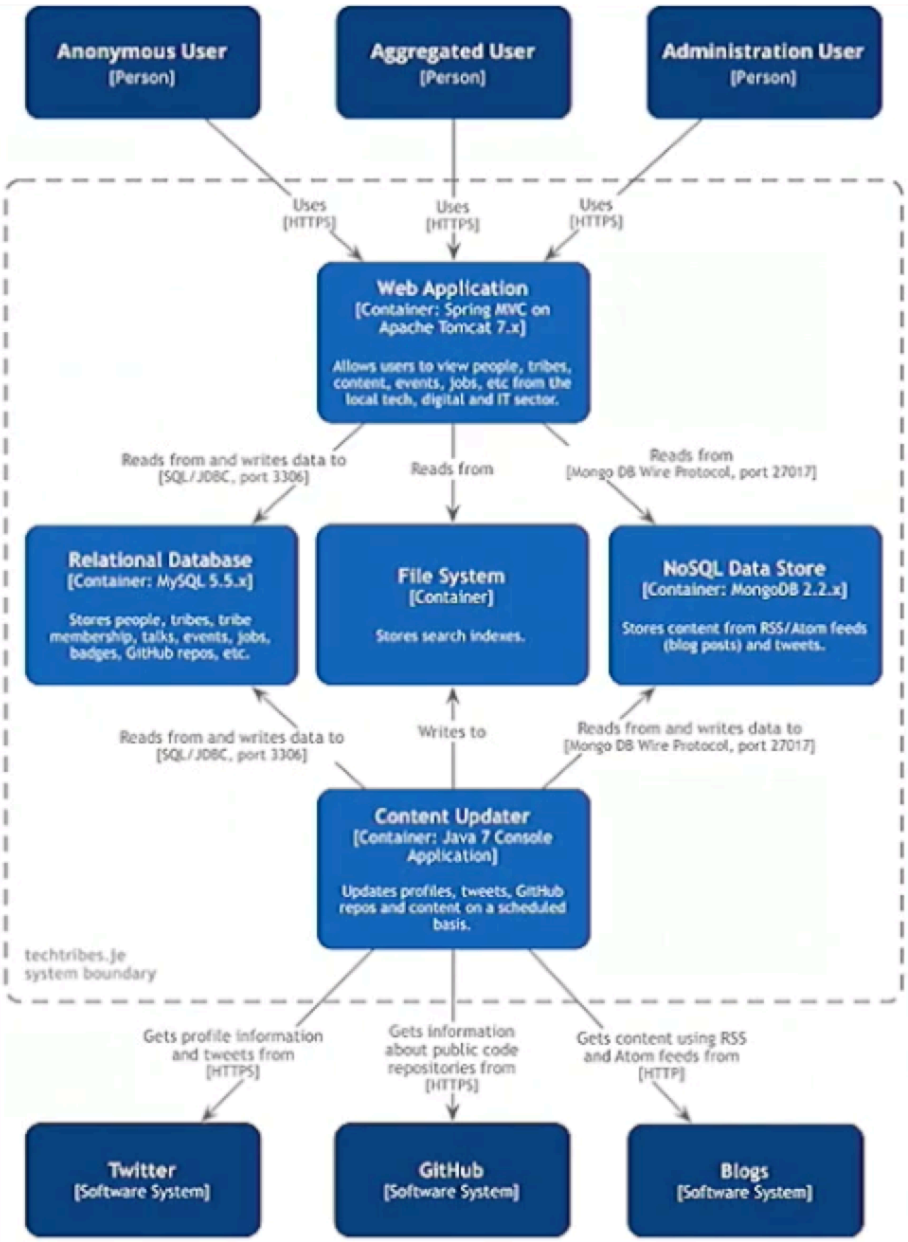
The component diagram for the API Application.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)


Anonymous User
Authenticated User
Admin User
[HTTP]
[HTTP]
[HTTP]
<<container>>
Web Server
Apache Tomcat 7.x
Allows users to view people, tribes, content, events, jobs, etc from the local tech, digital and IT sector.
Reads from and writes data to [SQL/JDBC, port 3306]
Reads from
Reads from and writes data to [Mongo DB Wire Protocol, port 27017]
<<container>>
Relational Database
MySQL 5.5.x
Stores people, tribes, tribe membership, talks, events, jobs, badges, GitHub repos, etc.
<<container>>
File System
Stores search indexes.
<<container>>
NoSQL Data Store
MongoDB 2.2.x
Stores content from RSS/ Atom feeds (blog posts) and tweets.
Reads from and writes data to [SQL/JDBC, port 3306]
Writes to
Reads from and writes data to [Mongo DB Wire Protocol, port 27017]
<<container>>
Content Updater
Standalone Java 7 process
Updates profiles, tweets, GitHub repos and content on a scheduled basis.
Gets data from [HTTP]
Gets data from [HTTP]
Gets data from [HTTP]
techtribes.je system boundary
<<external system>>
Twitter
<<external system>>
GitHub
<<external system>>
Blogs


<<container>>
Relational Database
MySQL 5.5.x
<<container>>
File System
<<container>>
NoSQL Data Store
MongoDB 2.2.x
Reads from and writes updated data to
techtribes.je
Content Updater
techtribes.je core components
(shown on a separate diagram)
Uses
<<component>>
Scheduled Content Updater
Spring Scheduled Task
Refreshes information from external systems every 15 minutes.
Uses
Uses
Uses
<<component>>
Twitter Connector
Twitter4J
Retrieves profile information and tweets (using the REST and Streaming APIs).
<<component>>
GitHub Connector
Eclipse Mylyn
Retrieves information about public repos.
<<component>>
News Feed Connector
ROME
Retrieves content from RSS and Atom feeds.
Gets data from [HTTP]
Gets data from [HTTP]
Gets data from [HTTP]
<<external system>>
Twitter
<<external system>>
GitHub
<<external system>>
Blogs

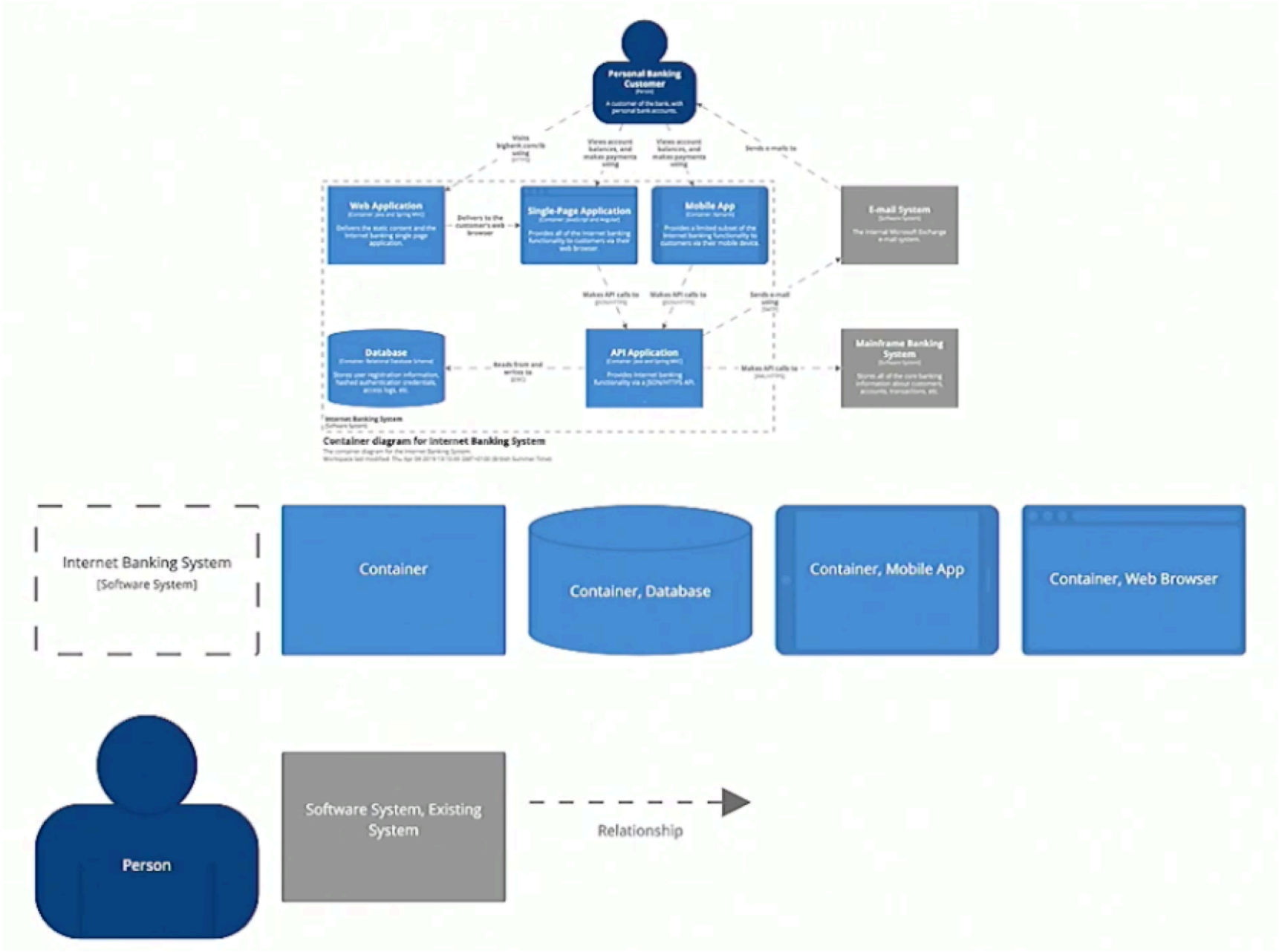

Anonymous User
Aggregated User
Administration User
Web Application
Relational Database
File System
NoSQL Data Store
Content Updater
techtribes.je
system boundary
Twitter
GitHub
Blogs

X


Anonymous User
[Person]
Aggregated User
[Person]
Administration User
[Person]
Uses
[HTTPS]
Uses
[HTTPS]
Uses
[HTTPS]
Web Application
[Container: Spring MVC on Apache Tomcat 7.x]
Allows users to view people, tribes, content, events, jobs, etc from the local tech, digital and IT sector.
Reads from and writes data to
[SQL/JDBC, port 3306]
Reads from
Reads from
[Mongo DB Wire Protocol, port 27017]
Relational Database
[Container: MySQL 5.5.x]
Stores people, tribes, tribe membership, talks, events, jobs, badges, GitHub repos, etc.
File System
[Container]
Stores search indexes.
NoSQL Data Store
[Container: MongoDB 2.2.x]
Stores content from RSS/Atom feeds (blog posts) and tweets.
Reads from and writes data to
[SQL/JDBC, port 3306]
Writes to
Reads from and writes data to
[Mongo DB Wire Protocol, port 27017]
Content Updater
[Container: Java 7 Console Application]
Updates profiles, tweets, GitHub repos and content on a scheduled basis.
techtribes.je
system boundary
Gets profile information and tweets from
[HTTPS]
Gets information about public code repositories from
[HTTPS]
Gets content using RSS and Atom feeds from
[HTTP]
Twitter
[Software System]
GitHub
[Software System]
Blogs
[Software System]

√

Uso de algunos íconos



Uso de íconos

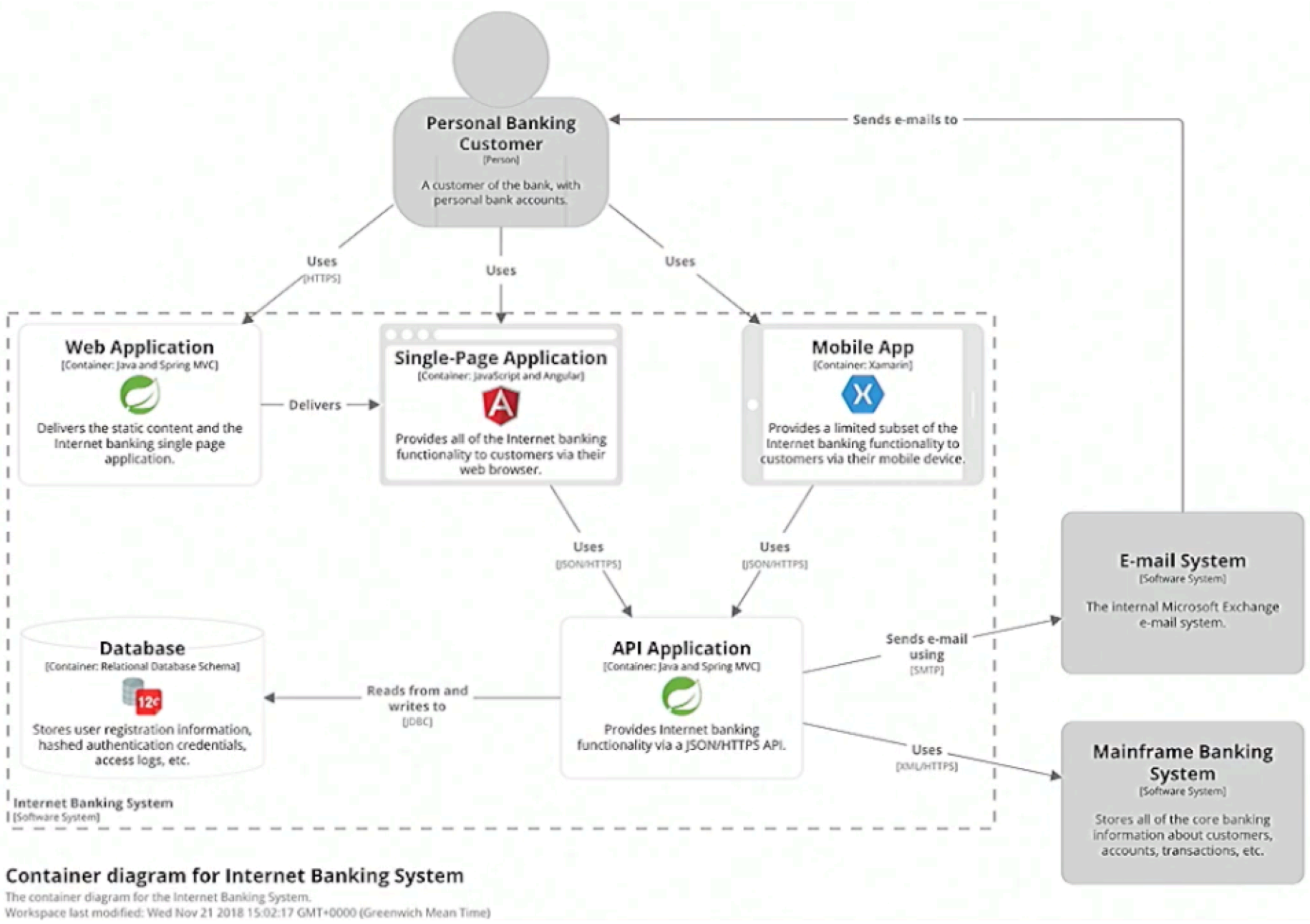
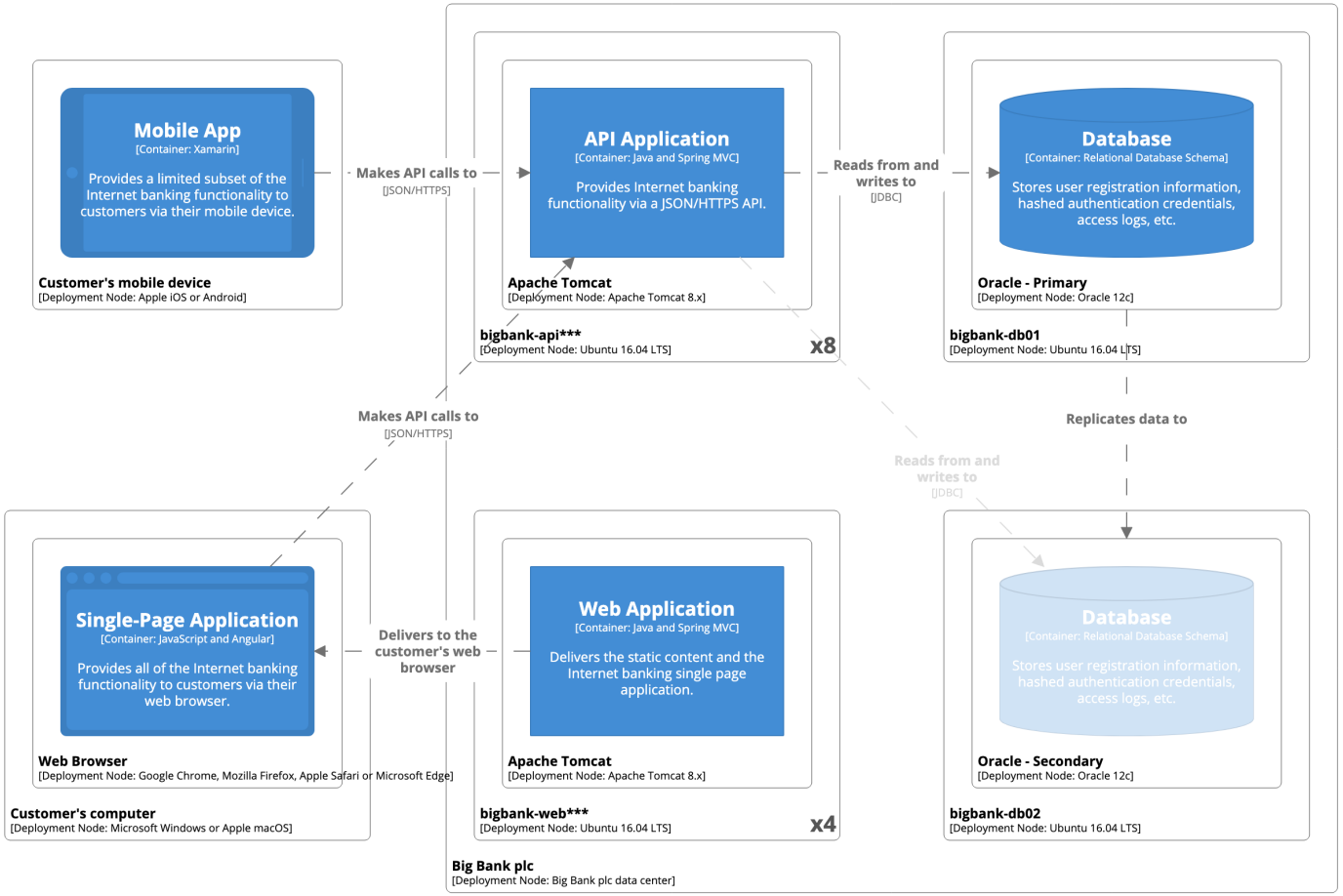


Diagrama de Despliegue

- Complementa el modelo C4, ilustra como los contenedores son mapeados a la infraestructura disponible



Deployment diagram for Internet Banking System - Live

An example live deployment scenario for the Internet Banking System.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)

Herramienta para Diagramar

Structurizr

Your software architecture documentation hub

Demo | Cloud service | On-premises installation | Sign up

Have you ever spent *hours* trying to make a software architecture diagram using a general purpose diagramming tool? Structurizr is a lightweight, web-based modelling tool that lets you quickly create diagrams based upon the C4 model for software architecture.

System Landscape diagrams | System Context diagrams | Container diagrams | Component diagrams | Dynamic diagrams | Deployment diagrams

Visualise

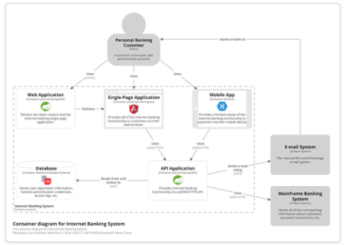
Create software architecture diagrams based upon the C4 model. Diagrams are interactive, navigable, and can be animated to help you tell stories during your presentations.

Document

Create supplementary documentation using Markdown or AsciiDoc, along with architecture decision records (ADRs) to capture your significant design decisions, all full-text searchable.

Explore

Explore your software architecture model from a number of perspectives, to get insight into the structure of your software.

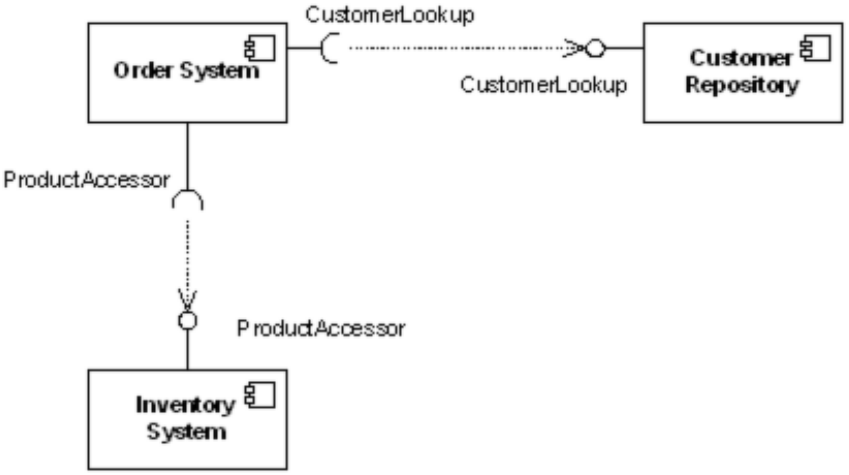


UML para Arquitectura

- Se puede describir la arquitectura en forma equivalente al modelo C4 usando diagramas de paquetes y componentes
- Diagrama de despliegue es un adicional

Diagrama UML de Componentes

Interfaz Provista *Interfaz Requerida*



- Un componente representa una parte modular de un sistema, que encapsula su contenido y cuya manifestación (el o los artefactos físicos correspondientes) es reemplazable dentro de su entorno.
- Un componente define su comportamiento en términos de interfaces provistas e interfaces requeridas

Diagrama UML de Componentes

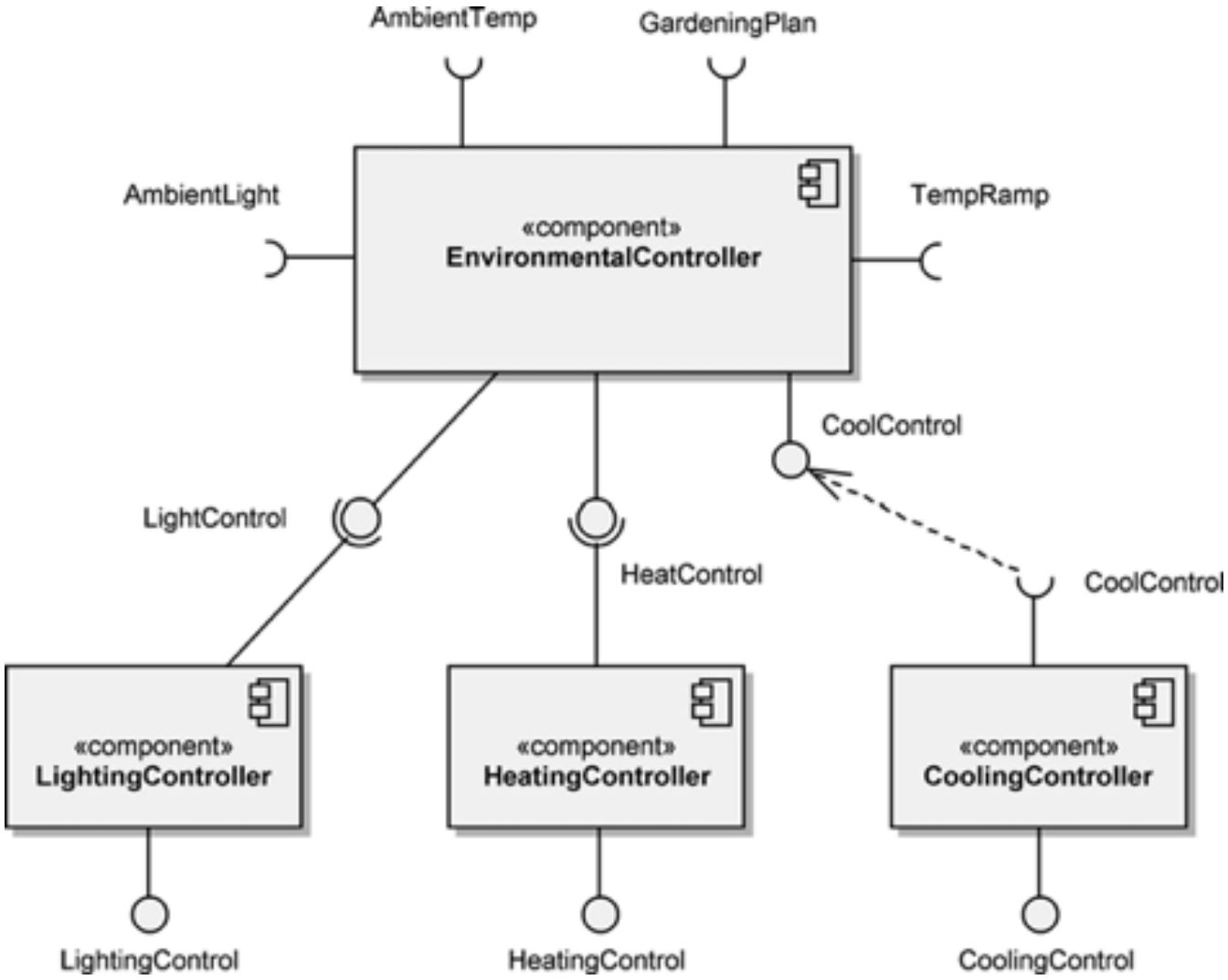


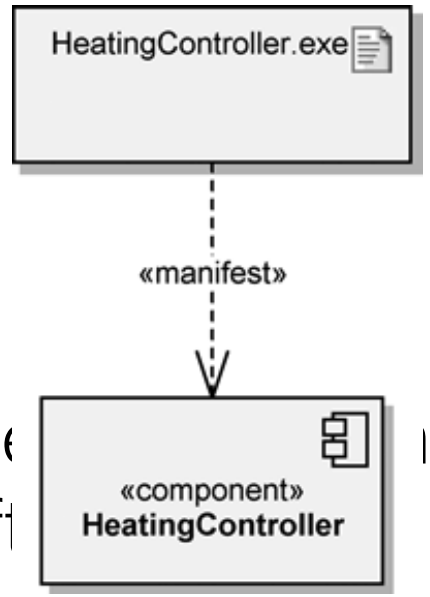
Diagrama de Despliegue (deployment)

- Muestra la asignación de artefactos de software concretos (archivos fuentes, archivos ejecutables, scripts y tablas de una base de datos) a nodos computacionales (algo que ofrece servicios de procesamiento).
- Es útil para comunicar la arquitectura física o de despliegue (instalación).

Artefactos y Nodos

Nodo: recurso computacional

- device (computer, modem, etc)
- execution environment (oracle engine, J2EE server)



Arte s físicos que implem e del soft

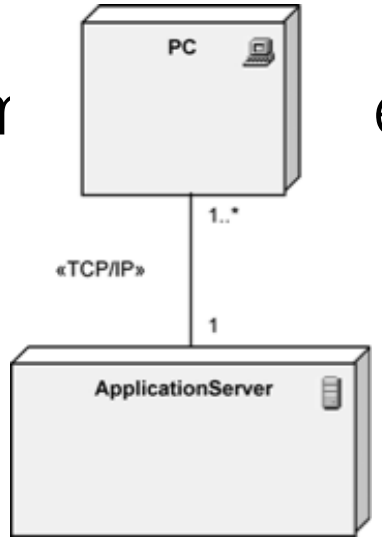
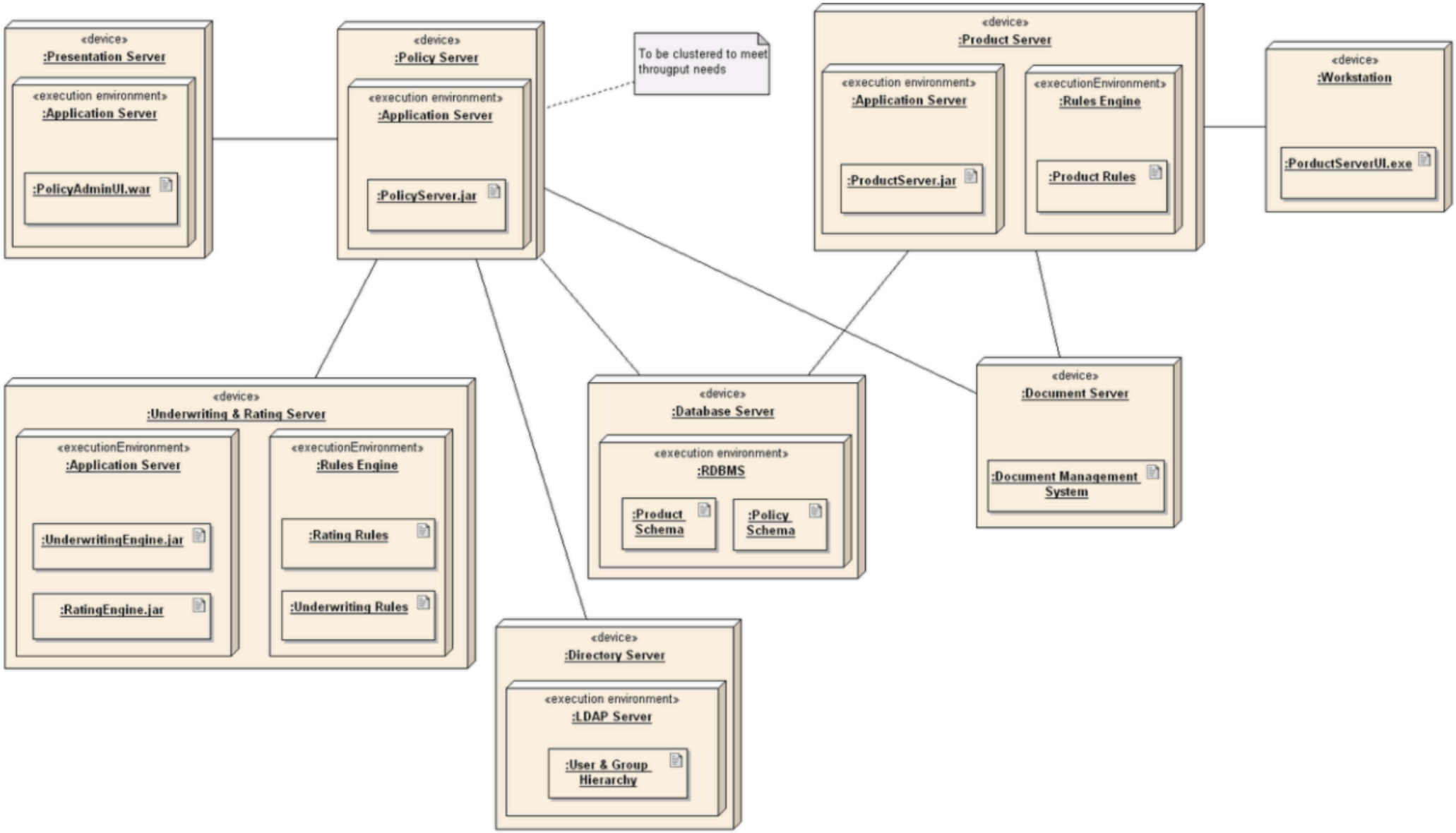
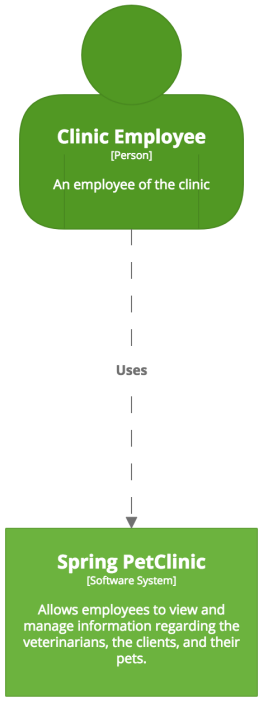
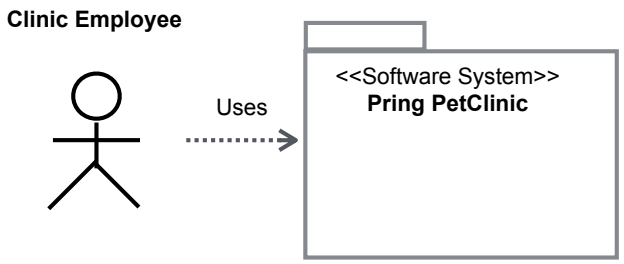


Diagrama UML de Despliegue



C4 vs UML (Context)



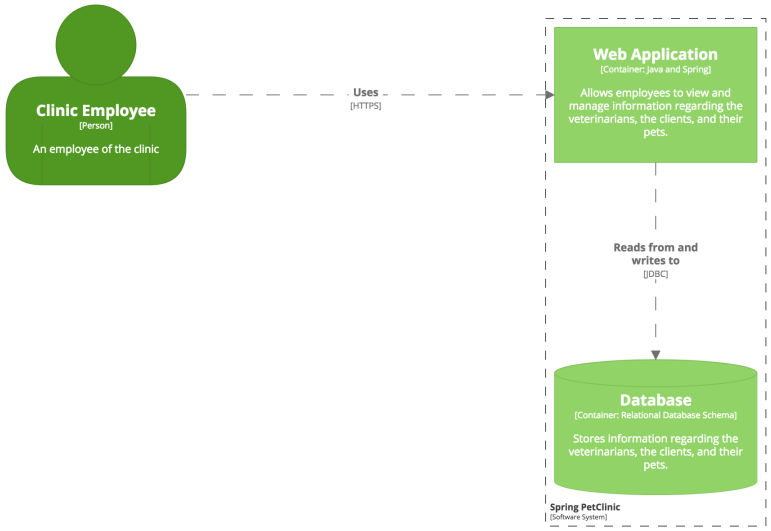


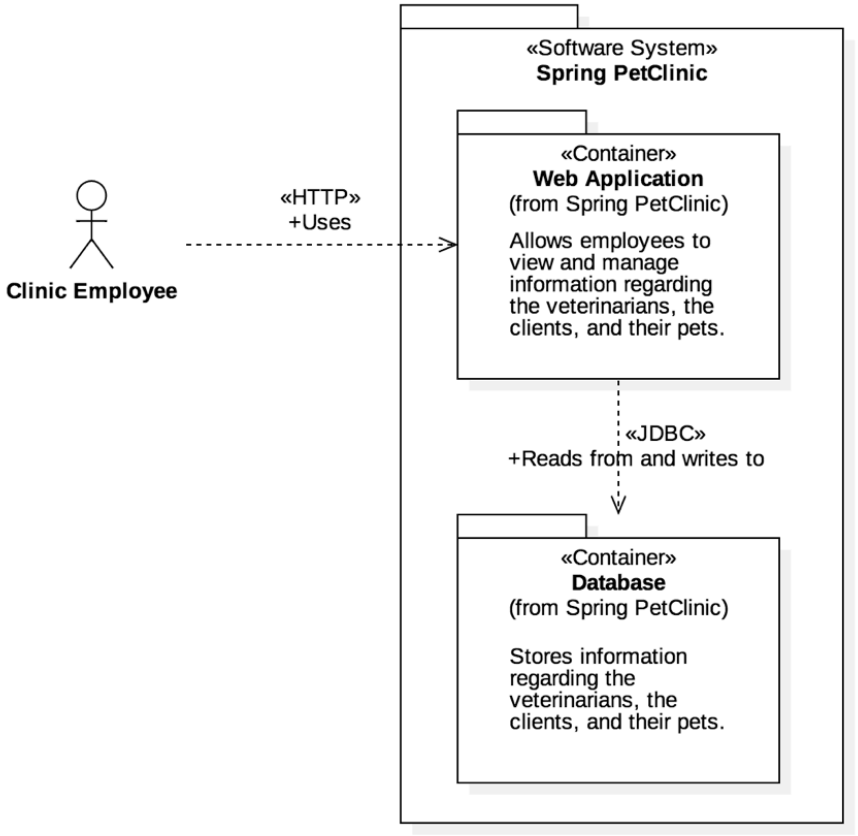
System Context diagram for Spring PetClinic

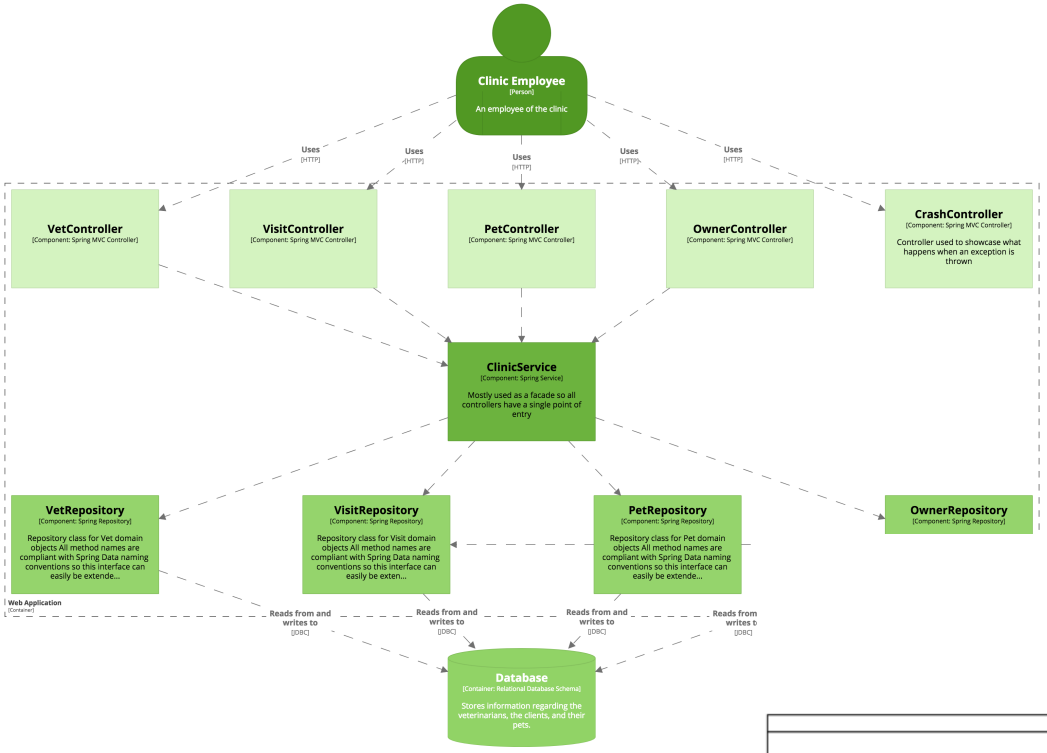
The System Context diagram for the Spring PetClinic system.

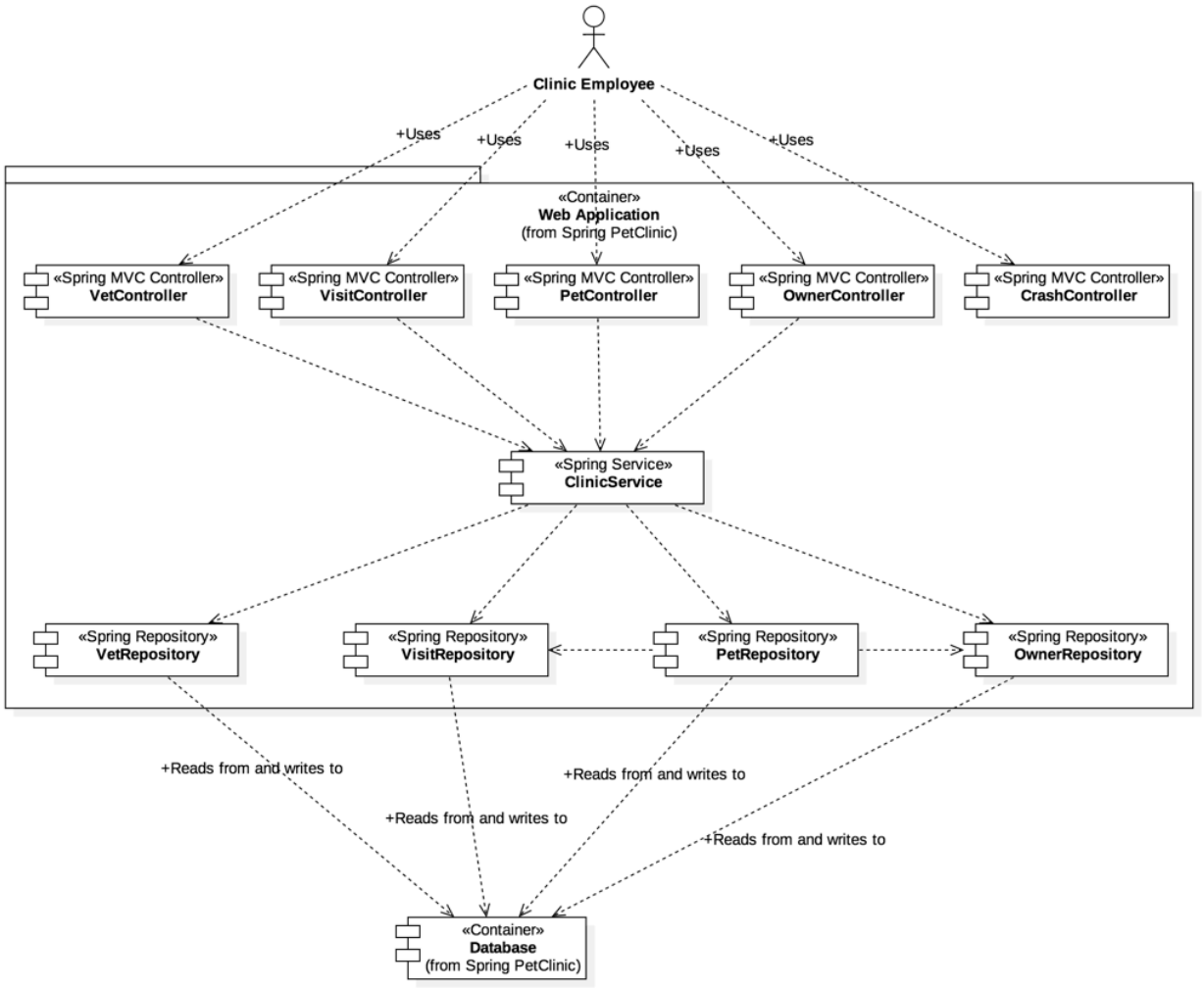
Last modified: Thursday 17 August 2017 10:15 UTC | Version: 95de1d9f8bf63560915331664b27a4a75ce1f1f6

C4 vs UML (Containers)






Clinic Employee
[Person]
An employee of the clinic
Uses
[HTTP]
VetController
[Component: Spring MVC Controller]
VisitController
[Component: Spring MVC Controller]
PetController
[Component: Spring MVC Controller]
OwnerController
[Component: Spring MVC Controller]
CrashController
[Component: Spring MVC Controller]
Controller used to showcase what happens when an exception is thrown
ClinicService
[Component: Spring Service]
Mostly used as a facade so all controllers have a single point of entry
VetRepository
[Component: Spring Repository]
Repository class for Vet domain objects All method names are compliant with Spring Data naming conventions so this interface can easily be extende...
VisitRepository
[Component: Spring Repository]
Repository class for Visit domain objects All method names are compliant with Spring Data naming conventions so this interface can easily be exten...
PetRepository
[Component: Spring Repository]
Repository class for Pet domain objects All method names are compliant with Spring Data naming conventions so this interface can easily be extende...
OwnerRepository
[Component: Spring Repository]
Web Application
[Container]
Reads from and writes to
[JDBC]
Database
[Container: Relational Database Schema]
Stores information regarding the veterinarians, the clients, and their pets.


Clinic Employee
+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
«Container»
Database
(from Spring PetClinic)