



**EDUCACIÓN**  
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO  
NACIONAL DE MÉXICO®



**Especialidad Análisis, diseño y desarrollo de software**

**D3.3 Diseño y documentación Modelo C4**

**Asesor: M.T.I.C. Leonardo Enriquez**

**Ingeniero electrónico, sistemas digitales**

# Modelo C4 para la arquitectura de software en microservicios

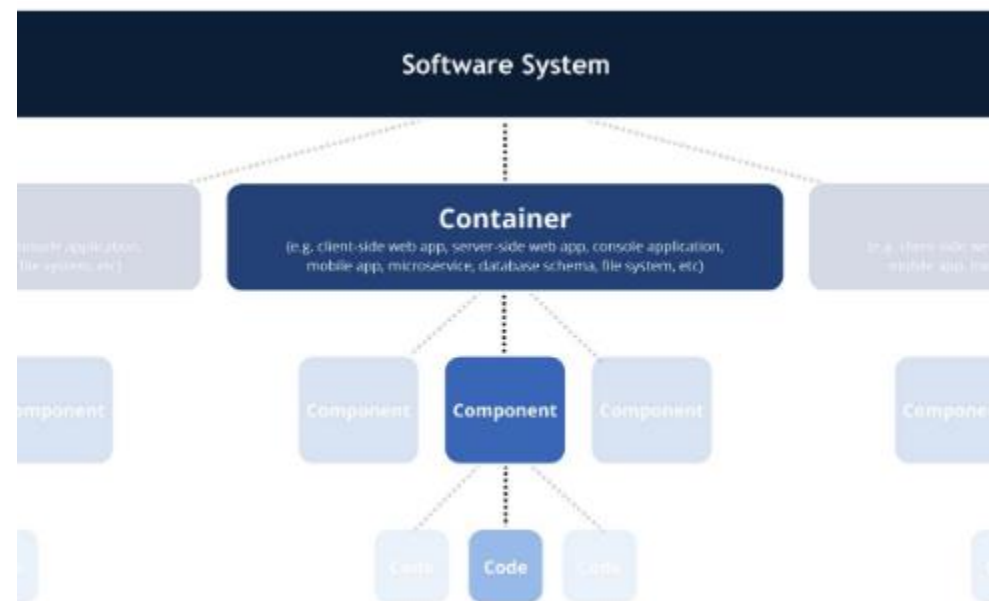
**Simon Brown** es un consultor independiente especializado en arquitectura de software y autor del libro “Software Architecture for Developers” (una guía amigable para desarrolladores de arquitectura de software, liderazgo técnico y equilibrio con agilidad). También es el creador del modelo de arquitectura de software C4, que es un enfoque simple para crear mapas de su código. Simon Brown es un orador habitual en conferencias internacionales de desarrollo de software y viaja por el mundo para ayudar a las organizaciones a visualizar y documentar su arquitectura de software.

## El “modelo C4”.

C4 significa contexto, contenedores, componentes y código — un conjunto de diagramas jerárquicos que puede utilizar para describir la arquitectura de su software en diferentes niveles de zoom, cada uno útil para diferentes audiencias.

## Puntos principales

- La creación de diagramas de software se redujo como resultado del cambio en el uso de metodologías ágiles. Cuando se crean diagramas, a menudo son confusos y poco claros.
- El modelo C4 consiste en un conjunto jerárquico de diagramas de arquitectura de software **para contexto, contenedores, componentes y código**.
- La jerarquía de los diagramas C4 proporciona diferentes niveles de abstracción, cada uno de los cuales es relevante para una audiencia diferente.
- Evite la ambigüedad en sus diagramas incluyendo una cantidad suficiente de texto, así como una clave o leyenda para la notación utilizada



# Modelo C4 perspectivas a través de 4 niveles



Like source code, Google Street View provides a very low-level and accurate view of a location.



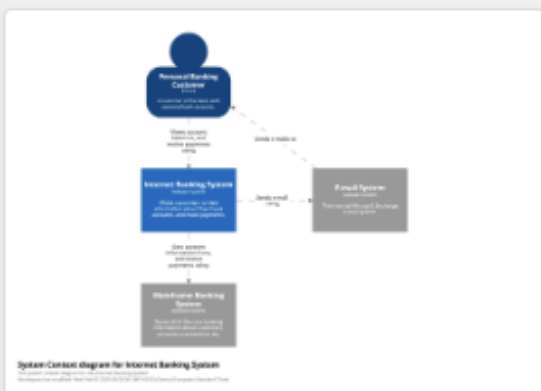
Navigating an unfamiliar environment becomes easier if you zoom out though.



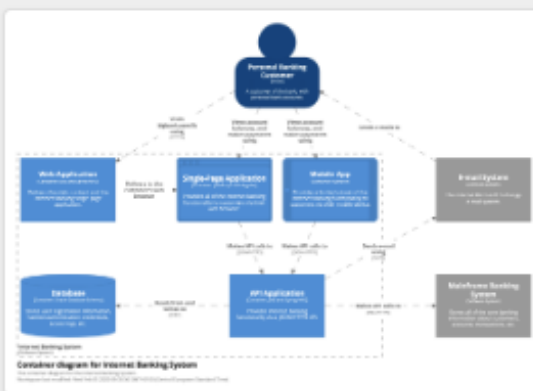
Zooming out further will provide additional context you might not have been aware of.



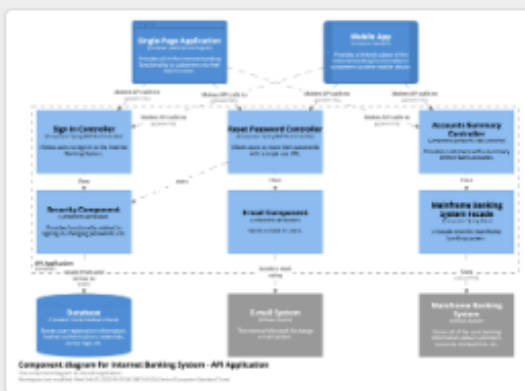
Different levels of zoom allow you to tell different stories to different audiences.



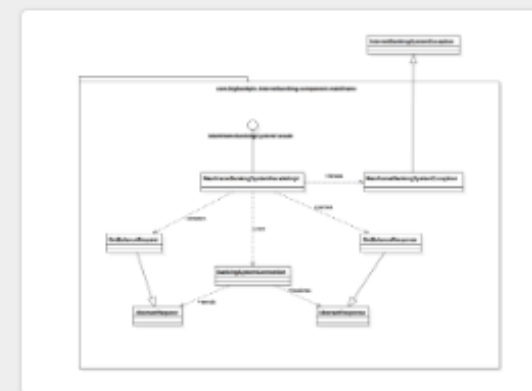
Level 1: A **System Context** diagram provides a starting point, showing how the software system in scope fits into the world around it.



Level 2: A **Container** diagram zooms into the software system in scope, showing the high-level technical building blocks.



Level 3: A **Component** diagram zooms into an individual container, showing the components inside it.

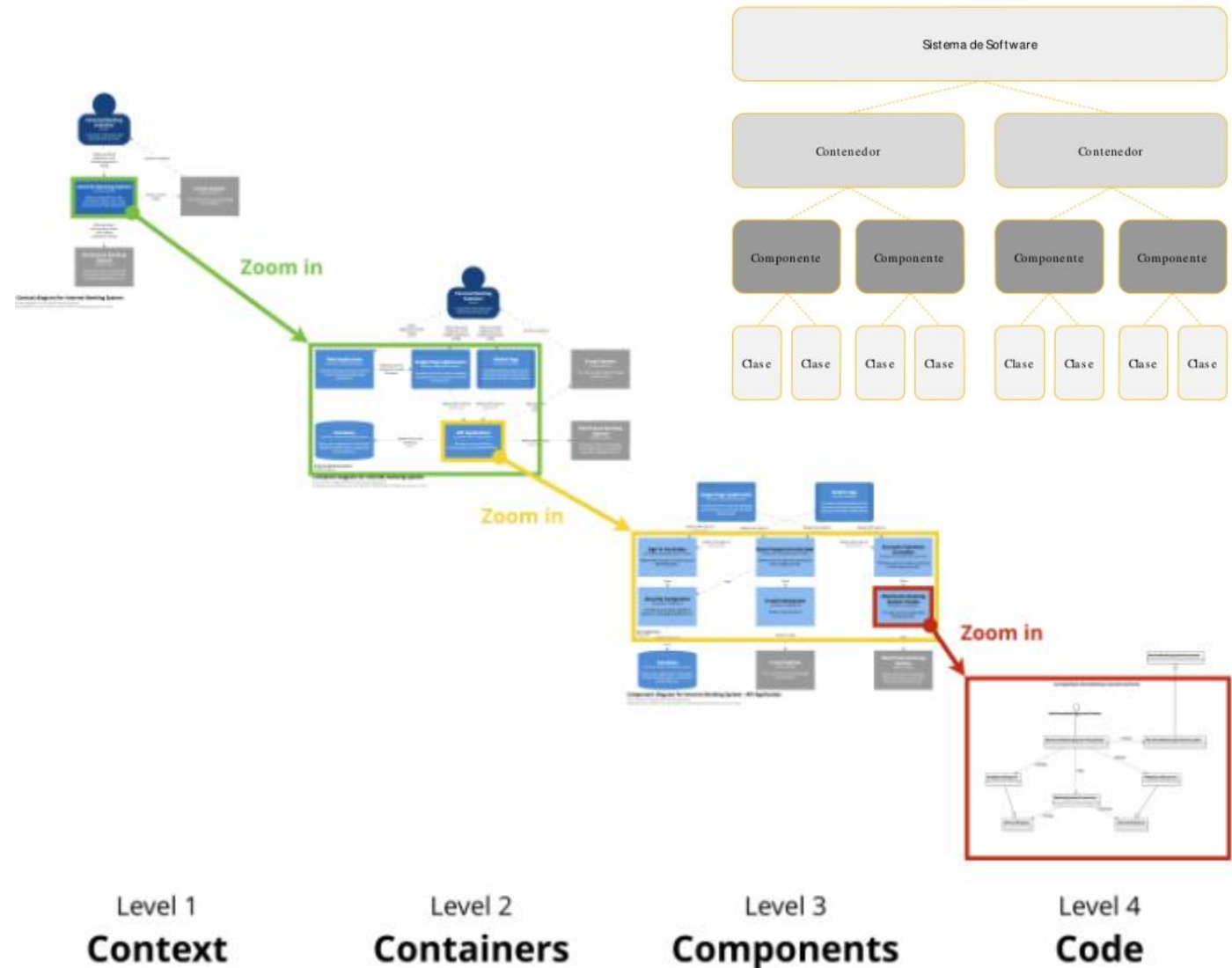


Level 4: A **code** (e.g. UML class) diagram can be used to zoom into an individual component, showing how that component is implemented.

# Modelo C4 definición de los 4 niveles de abstracción

Una metodología para la representación de diagramas que he estado poniendo en práctica recientemente es el modelo C4 por Simon Brown, en el cual, siguiendo ciertas prácticas e iterando sobre nuestros diagramas, podemos diseñar nuestro sistema utilizando varios niveles de abstracción:

1. **Contexto:** Visualiza el sistema en la organización, su relación con distintos actores que pueden ser usuarios y otros sistemas.
2. **Contenedores:** Visualiza la división del sistema en entornos de ejecución o almacenamiento, reflejando mejor la infraestructura.
3. **Componentes:** Visualiza internamente las piezas que componen un contenedor en particular, como aplicaciones o servicios.
4. **Clases:** Consiste en un diagrama de clases UML para aquellos casos donde sí sea necesario llegar a ese nivel.



# Modelo C4 anotaciones propuestas

## Anotaciones

El modelo C4 presenta los diagramas con notaciones simple que funciona bien en pizarras, papeles, notas adhesivas, fichas y una variedad de herramienta de diseño. También puede utilizar UML como anotación, con el uso apropiado de paquetes, componentes y estereotipos.

Independiente de la notación que se utilice, se recomienda que cada elemento incluya nombre, tipo de elemento (es decir, “Persona”, “Sistema de software”, “Contenedor” o “Componente”), una opción tecnología (si procede) y algún texto descriptivo.

Asegure de tener una clave o leyenda que describa cualquier notación que se este utilizado, inclusive si es obvio para usted. Se puede cubrir colores, formas, acrónimos, estilos de líneas. Lo ideal es que la anotación sea coherente en todos los niveles de detalle.



<https://c4model.com/>



# Modelo C4 Elementos y relaciones

## Elements and relationships

Element type	Parent	Properties
Person	None	<ul style="list-style-type: none"><li>Name*</li><li>Description</li><li>Location (Internal or External)</li></ul>
Software System	None	<ul style="list-style-type: none"><li>Name*</li><li>Description</li><li>Location (Internal or External)</li><li>The set of containers that make up the software system</li></ul>
Container	A software system	<ul style="list-style-type: none"><li>Name*</li><li>Description</li><li>Technology</li><li>The set of components within the container</li></ul>
Component	A container	<ul style="list-style-type: none"><li>Name*</li><li>Description</li><li>Technology</li><li>The set of code elements (e.g. classes, interfaces, etc) that the component is implemented by</li></ul>
Code Element	A component	<ul style="list-style-type: none"><li>Name*</li><li>Description</li><li>Fully qualified type</li></ul>
Relationship**		<ul style="list-style-type: none"><li>Description</li><li>Technology</li></ul>

\* All elements in the model must have a name, and that name must be unique within the parent context.

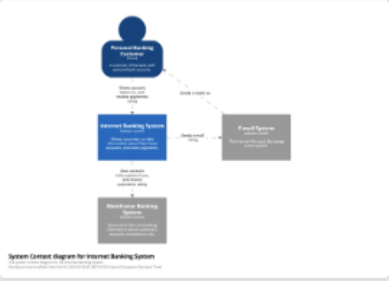



\*\* Relationships are permitted between any elements in the model, in either direction.

## Elementos y relaciones modelo C4

- ✓ **Personas**
  - ✓ Padre:
  - ✓ Propiedades:
    - ✓ Nombre
    - ✓ Descripción
    - ✓ Locación (interna o externa)
- ✓ **Sistema de software**
  - ✓ Padre:
  - ✓ Propiedades:
    - ✓ Nombre
    - ✓ Descripción
    - ✓ Locación (interna o externa)
    - ✓ Contenedores del sistema
- ✓ **Contenedor**
  - ✓ Padre: Sistema de software
  - ✓ Propiedades:
    - ✓ Nombre
    - ✓ Descripción
    - ✓ Tecnología
    - ✓ Componentes del contenedor
- ✓ **Componente**
  - ✓ Padre: Contenedor
  - ✓ Propiedades:
    - ✓ Nombre
    - ✓ Descripción
    - ✓ Tecnología
    - ✓ Elementos del código (clases, interfaces)
- ✓ **Elemento del código**
  - ✓ Padre: Componente
  - ✓ Propiedades:
    - ✓ Nombre
    - ✓ Descripción
    - ✓ Tipo de calificador
- ✓ **Relación**
  - ✓ Propiedades:
    - ✓ Descripción
    - ✓ Tecnología

# Modelo C4 Vistas, alcances y elementos permitidos

## Views

View type	Scope	Permitted elements	
1. System Context	A software system.	<ul style="list-style-type: none"><li>Software systems</li><li>People</li></ul>	
2. Container	A software system	<ul style="list-style-type: none"><li>Software systems</li><li>People</li><li>Containers within the software system in scope</li></ul>	
3. Component	A container	<ul style="list-style-type: none"><li>Software systems</li><li>People</li><li>Other containers within the parent software system of the container in scope</li><li>Components within the container in scope</li></ul>	
4. Code	A component	<ul style="list-style-type: none"><li>Code elements (e.g. classes, interfaces, etc) that are used to implement the component in scope</li></ul>	

### Vistas del modelo C4

- ✓ **1. Vista de contexto del sistema**
  - ✓ Alcance: Un sistema de software
  - ✓ Elementos permitidos:
    - ✓ Sistema de software
    - ✓ Personas
- ✓ **2. Vista contenedor**
  - ✓ Alcance: Un sistema de software
  - ✓ Elementos permitidos:
    - ✓ Sistema de software
    - ✓ Personas
    - ✓ Contenedor dentro del alcance del software
- ✓ **3. Vista Componente**
  - ✓ Alcance: Contenedor
  - ✓ Elementos permitidos:
    - ✓ Sistema de software
    - ✓ Personas
    - ✓ Otros contenedor dentro del sistema de software padre
    - ✓ Componentes dentro del contenedor padre
- ✓ **4. Vista Código**
  - ✓ Alcance: Componente
  - ✓ Elementos permitidos:
    - ✓ Elementos del Código (clases, interfaces, etc), usadas en la implementación del componente



# Modelo C4 Nivel 1 Contexto del sistema

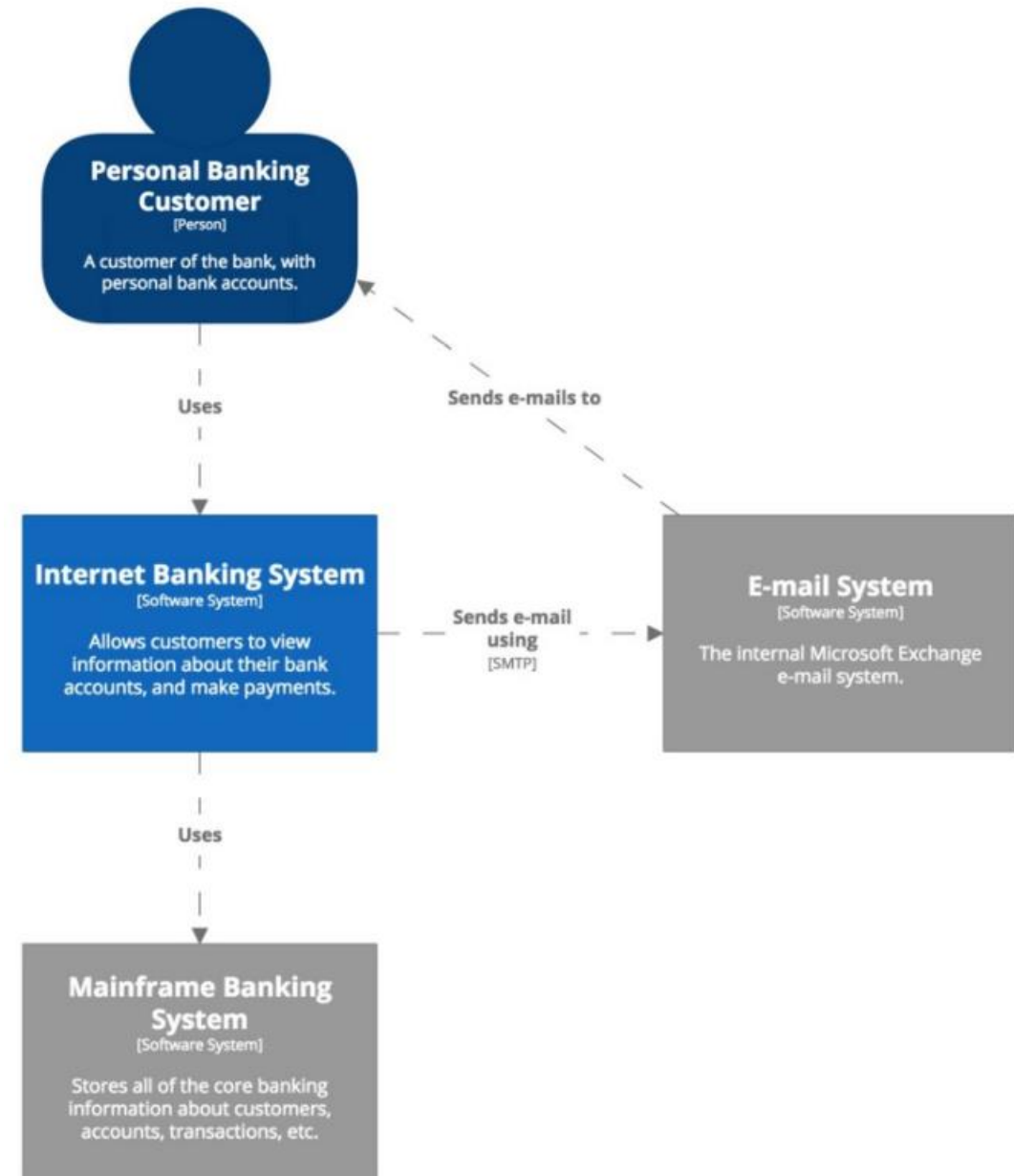
## Nivel 1: El diagrama de contexto del sistema

Este nivel muestra el sistema de software que está construyendo y cómo encaja en el mundo en términos de las personas que lo utilizan y los otros sistemas de software con los que interactúa.

A que se define quienes son los actores que interactúan con nuestro sistema, qué sistemas tenemos, y una primera idea de cómo se relacionan entre ellos. Es también una oportunidad para destacar qué queda fuera del sistema, y hasta qué punto se pueden modificar esos sistemas. Podemos utilizar esquemas de colores para destacar qué sistemas podemos y cuales no podemos modificar.

Aquí hay un ejemplo de un diagrama de contexto que describe un sistema de banca por Internet que usted puede estar construyendo:

- Sus clientes bancarios utilizan el sistema de banca por Internet para visualizar información sobre sus cuentas bancarias y realizar pagos.
- El sistema de banco por internet utiliza el sistema de correo existente del banco para enviar correo electrónico a los clientes.
- El código de color en el diagrama indica los sistemas de software que ya existen (cajas grises) y los que se van a construir (azules)





# Modelo C4 Nivel 2 Contenedor del sistema

## Nivel 2: El diagrama del contenedor

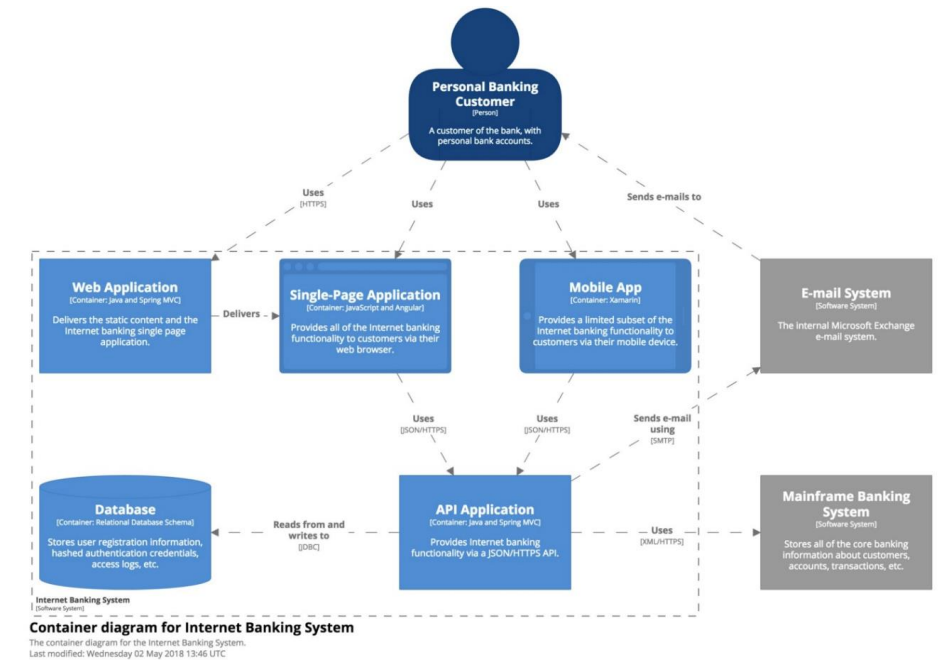
Un diagrama de contenedor, amplía el sistema de software y muestra los contenedores (**aplicaciones, almacenamiento de datos, microservicios, etc.**) que componen este sistema de software. Las decisiones tecnológicas son también una parte fundamental de este diagrama.

En este caso hacemos zoom en cada uno de los sistemas, y lo separamos en contenedores que tienen una función lógica. En el caso de nuestro blog tenemos dos subsistemas diferentes, uno encargado de la edición de artículos y otro de la visualización, así como una API entre la que interactúan ambos sistemas.

A este nivel podemos tener conversaciones a nivel técnico con diferentes equipos involucrados en el proyecto, el nivel de abstracción es menor, y podemos tomar decisiones como cuantos subsistemas construimos y cómo se reparte el trabajo de los mismos.

El ejemplo es un diagrama de contenedor para el sistema de banca por Internet. Esto muestra que el sistema de banca por internet (el cuadro punteado) consta de cinco contenedor:

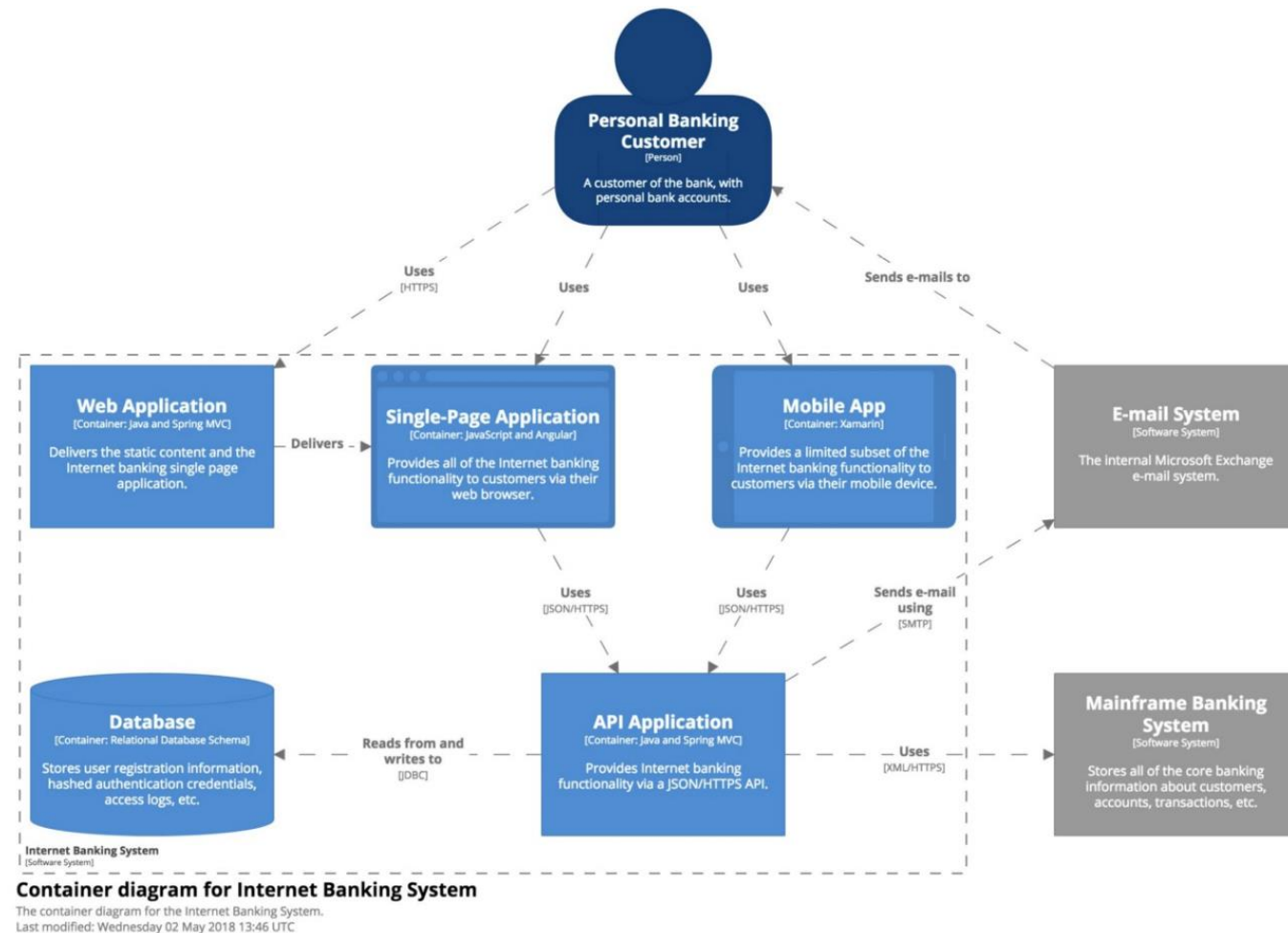
- Una aplicación web del lado del servidor,
- Una aplicación de una sola pagina del lado del cliente,
- Una aplicación móvil,
- Una aplicación API del lado del servidor
- Y una base de datos.



# Modelo C4 Nivel 2 Contenedor del sistema (explicación a través de un ejemplo)

## Nivel 2: El diagrama del contenedor

La aplicación Web es un sistema Java/Spring MVC que simplemente muestra contenido estático (HTML, CSS y Javascript), incluyendo el contenido que constituye la aplicación de una sola pagina. La aplicación de una sola pagina en una aplicación Angular que se ejecuta en el navegador web del cliente, proporcionando todas las características de la banca por Internet. Alternativamente, los cliente pueden utilizar un app móvil en Xamarin. La aplicación de una sola pagina y la aplicación móvil utilizan una API JSON/HTTPS, que proporciona otra aplicación Java/Spring MVC que se ejecuta en el lado del servidor. La aplicación en la API obtiene información de usuario de la base de datos (un esquema de base de datos relacional. La aplicación de la API también se comunica con el sistema bancario en el mainframe existente, utilizando una interfaz XML/HTTPS propietaria, para realizar transacciones. La aplicación de la API también utiliza el sistema de correo electrónico existente sin necesita enviar correo electrónico a los clientes.



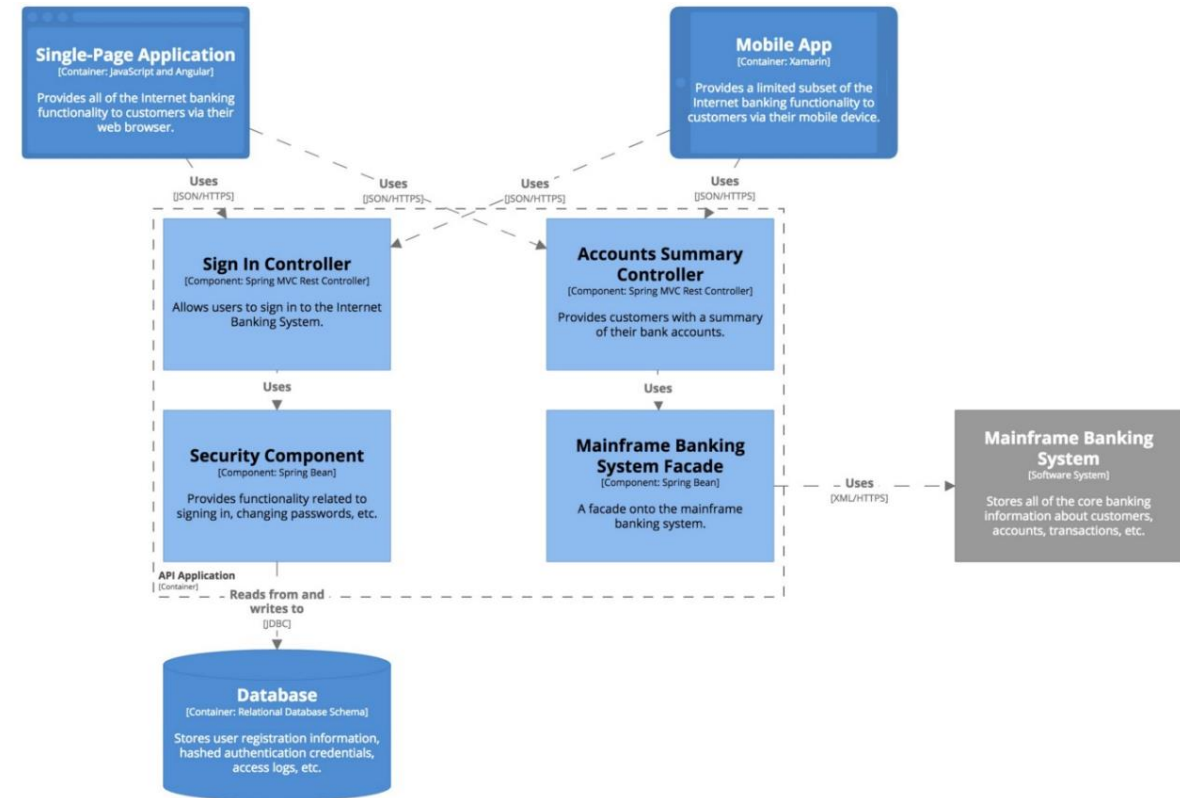
# Modelo C4 Nivel 3 Componentes del sistema

## Nivel 3: El diagrama de componentes

El diagrama de componentes, expande un contenedor individual para mostrar los componentes que contiene. Estos componentes deben asignarse a abstracciones reales (por ejemplo, una agrupación de códigos) en función de su código.

Este nivel es el más profundo al que llegaremos en este artículo, aquí podemos establecer una separación más clara, en el caso de la UI podemos separar las diferentes páginas que existirán, y en el caso de la API backend los diferentes componentes que podemos emplear, en el que cada componente puede ser una librería, un paquete, un proyecto o una unidad semántica de código.

En este nivel podemos tomar decisiones de arquitectura directamente a nivel de equipo, entender las diferentes relaciones entre los diferentes componentes y modularizar más, si fuese necesario.



**Component diagram for Internet Banking System - API Application**

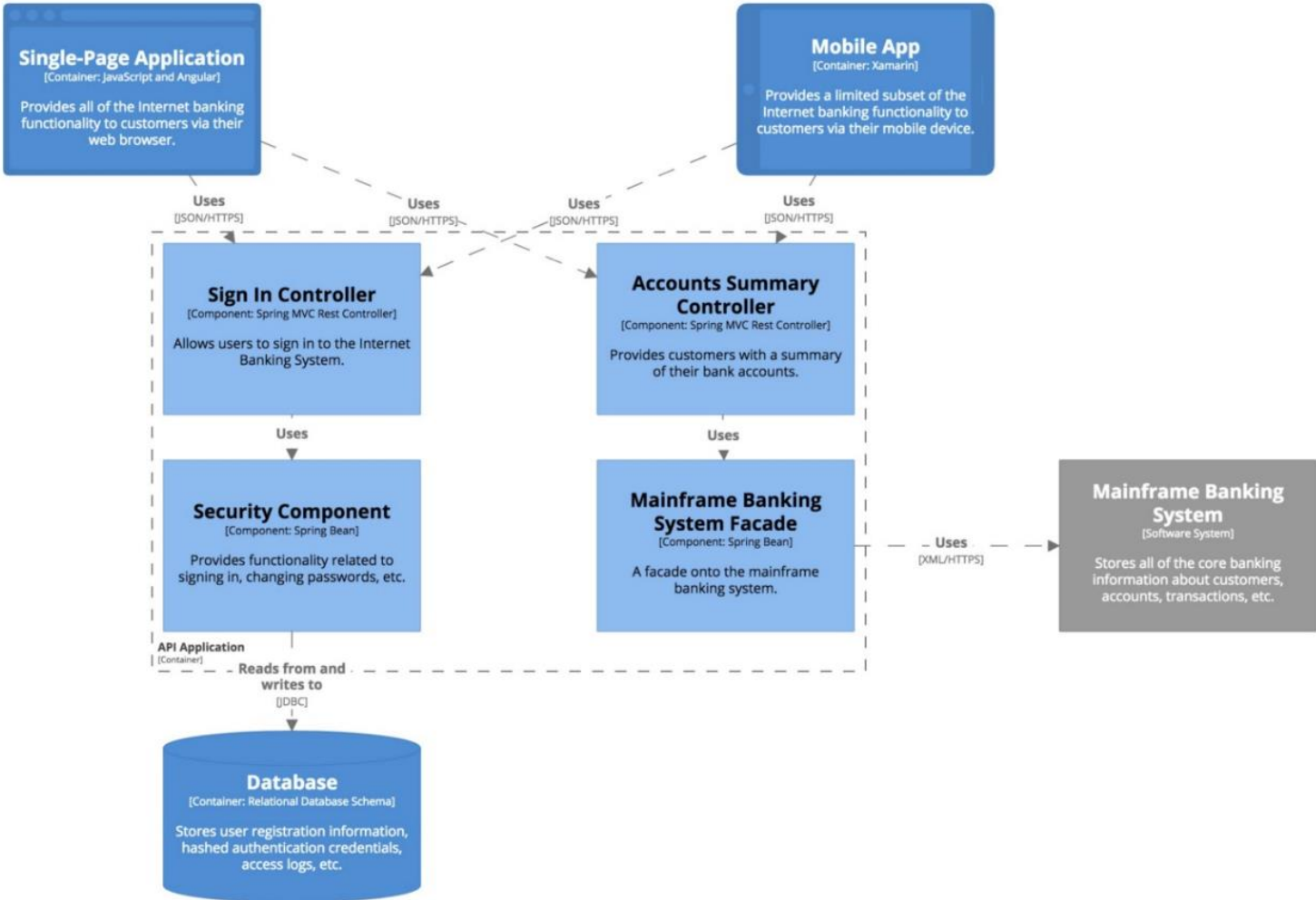
The component diagram for the API Application.  
Last modified: Wednesday 02 May 2018 13:46 UTC

# Modelo C4 Nivel 3 Componentes del sistema (explicación a través de un ejemplo)

## Nivel 3: El diagrama de componentes

En el ejemplo se muestra un diagrama de componentes para el sistema ficticio de banca por Internet que muestra algunos de los componentes de la aplicación API.

Dos controladores MVC Spring en **Rest** proporcionan punto de acceso a la **API de JSON/HTTPS**, y cada controlador utiliza posteriormente otros componentes para acceder a los datos de la base de datos y del sistema bancario en el mainframe.



Component diagram for Internet Banking System - API Application

The component diagram for the API Application.  
Last modified: Wednesday 02 May 2018 13:46 UTC

# Modelo C4 Nivel 4 Clases del sistema

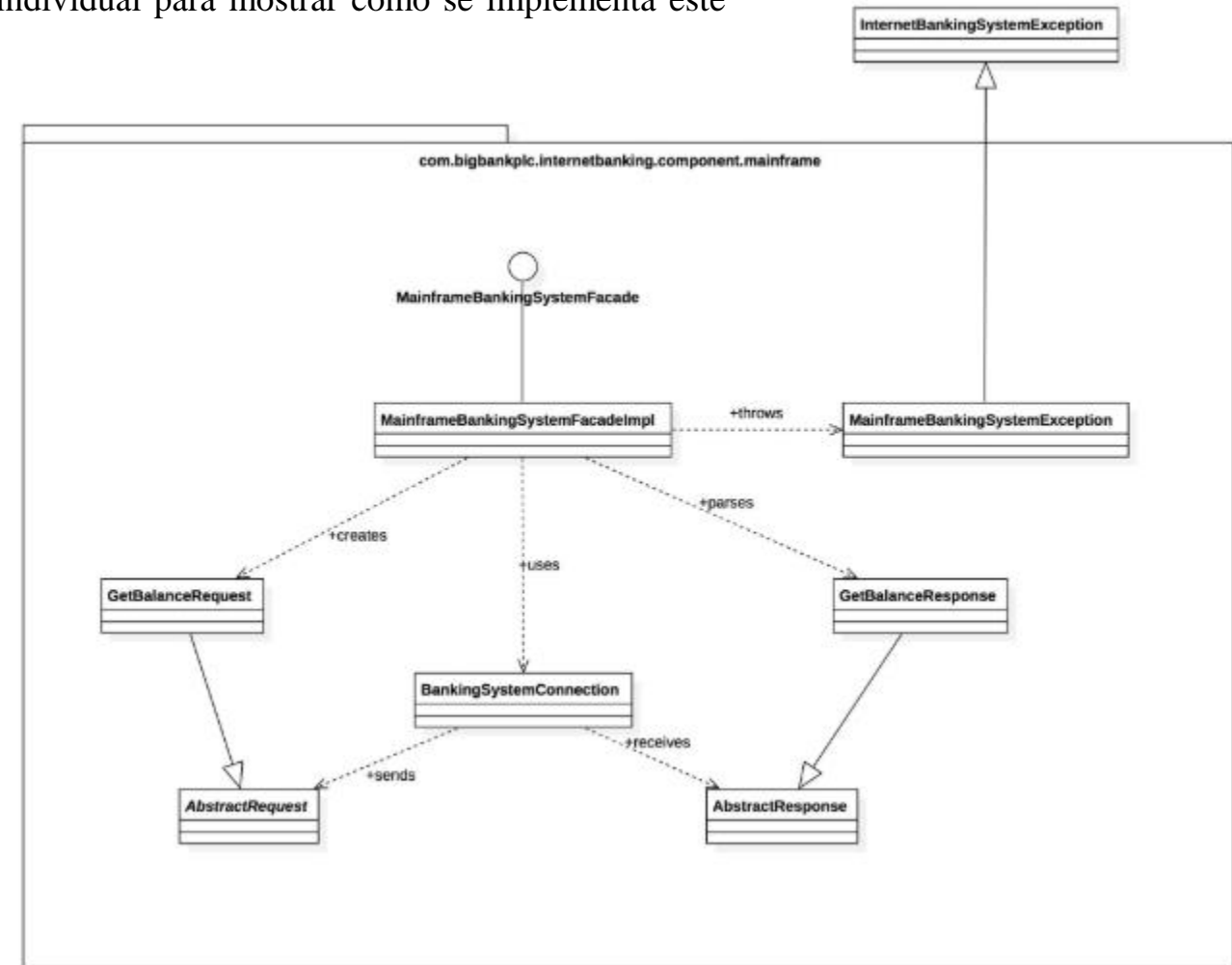
## Nivel 4: El código o clases

Por último, si realmente lo desea o necesita, puede ampliar un componente individual para mostrar cómo se implementa este componente.

Este diagrama muestra que el componente consta de varias clases y que los detalles de implementación reflejan directamente el indicador. No recomendaría necesariamente la creación de diagramas a este nivel de detalle, especialmente cuando se pueden obtener bajo demanda de la mayoría de los IDEs.

Este es un diagrama de ejemplo (y parcial) de **clases UML** para el sistema ficticio de banca por Internet que muestra los elementos de código (interfaces y clases) que componen el componente MainframeBankingSystemFacade.

Este diagrama muestra que el componente consta de varias clases y que los detalles de implementación reflejan directamente el indicador. No se recomienda necesariamente la creación de diagramas a este nivel de detalle, especialmente cuando se pueden obtener bajo demanda de la mayoría de los IDEs.



# Modelo C4 Nivel 1 Diagrama de contexto (Ejemplo)

## Nivel 1 El diagrama de contexto del sistema

La vista 1 es la de contexto del sistema tiene como objetivo, visualizar nuestro sistema desde un alto nivel, viendo lo que vamos a desarrollar en el centro rodeado de todos los sistemas y usuarios que interactúan con él. En este nivel los detalles no son importantes, no hay que preocuparse por protocolos ni mecanismos de comunicación. Lo importante es centrarse en las interacciones con el exterior.

El siguiente diagrama muestra el nivel de contexto para el ejemplo que cuestión.

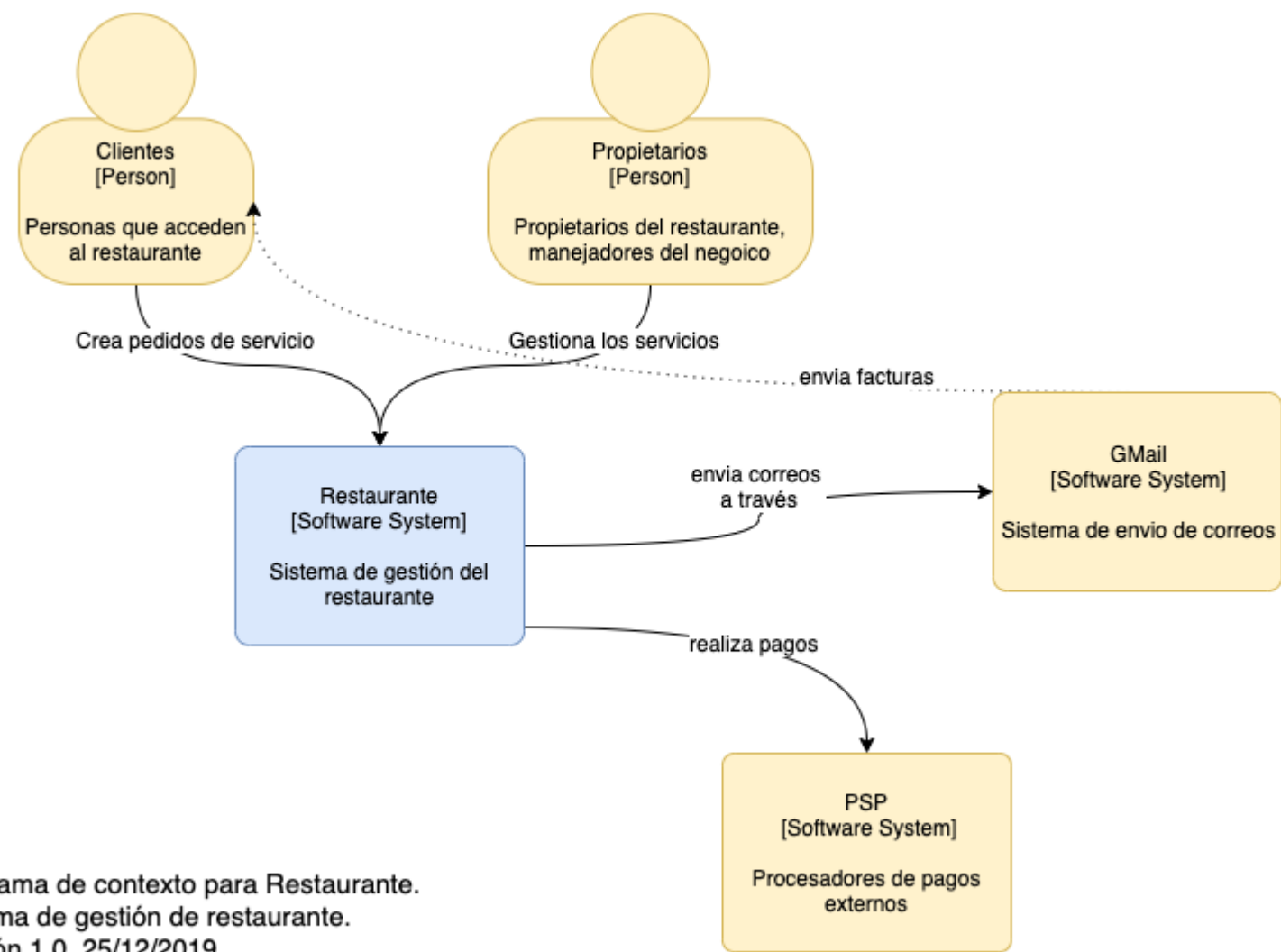


Diagrama de contexto para Restaurante.  
Sistema de gestión de restaurante.  
Versión 1.0, 25/12/2019



# Modelo C4 Nivel 2 Diagrama de contenedor (Ejemplo)

## Nivel 2 El diagrama del contenedor

La vista de contenedores es un zoom, una mirada hacia adentro de lo que vamos a desarrollar, es en esencia una ampliación del sistema a desarrollar, ilustrando los elementos que lo contienen y sus interacciones. Se especifican todos los elementos a desarrollar y usados de terceros como pueden ser sistemas de bases de datos además de sus interacciones.

En el ámbito de los microservicios serás cada uno de los sistemas que exponen APIs, las bases de datos, y los elementos propios de soporte a la infraestructura como pueden ser, el servicio de configuración centralizada, descubrimiento, gateway, etc.

El siguiente gráfico muestra el diagrama de contenedores para el sistema de Restaurante que estamos modelando, no se han incluido los servicios de soporte a la infraestructura.

El siguiente gráfico muestra el diagrama de contenedores para el sistema de Restaurante que estamos modelando, no se han incluido los servicios de soporte a la infraestructura.

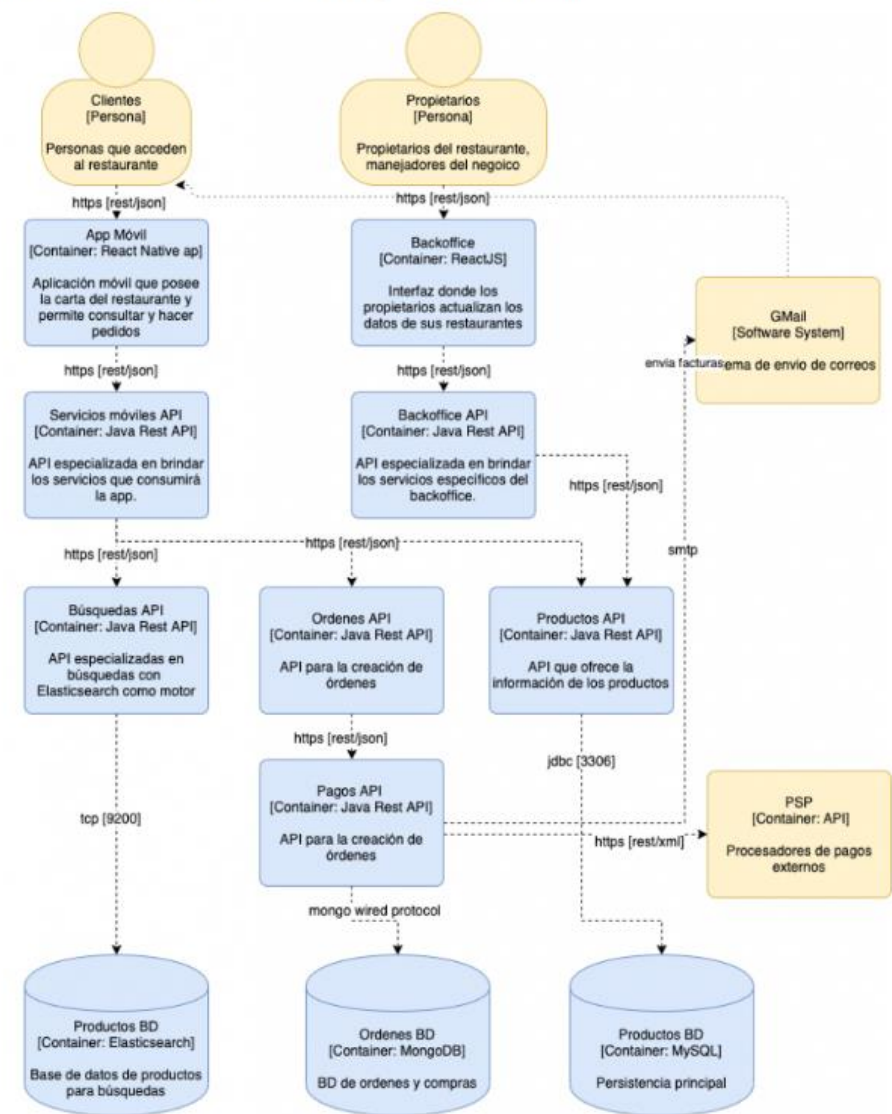


Diagrama de contenedores para Restaurante.  
Sistema de gestión de restaurante.  
Versión 1.0, 25/12/2019

# Modelo C4 Nivel 3 Diagrama de componentes (Ejemplo)

## Nivel 3 Componentes del sistema

Similar al nivel anterior, el nivel de componentes del sistema es una vista ampliada de cada uno de los contenedores del sistema que vamos a desarrollar. No incluye un diagrama detallado de componentes de terceros que reusamos como las bases de datos, solo de los sistemas que vamos a desarrollar y se realiza un diagrama por cada componente que se desee modelar.

El siguiente diagrama es para el contenedor BackOffice API que hemos extraído del diagrama anterior y queremos explicar sus componentes internos.

El siguiente diagrama es para el contenedor BackOffice API que hemos extraído del diagrama anterior y queremos explicar sus componentes internos.

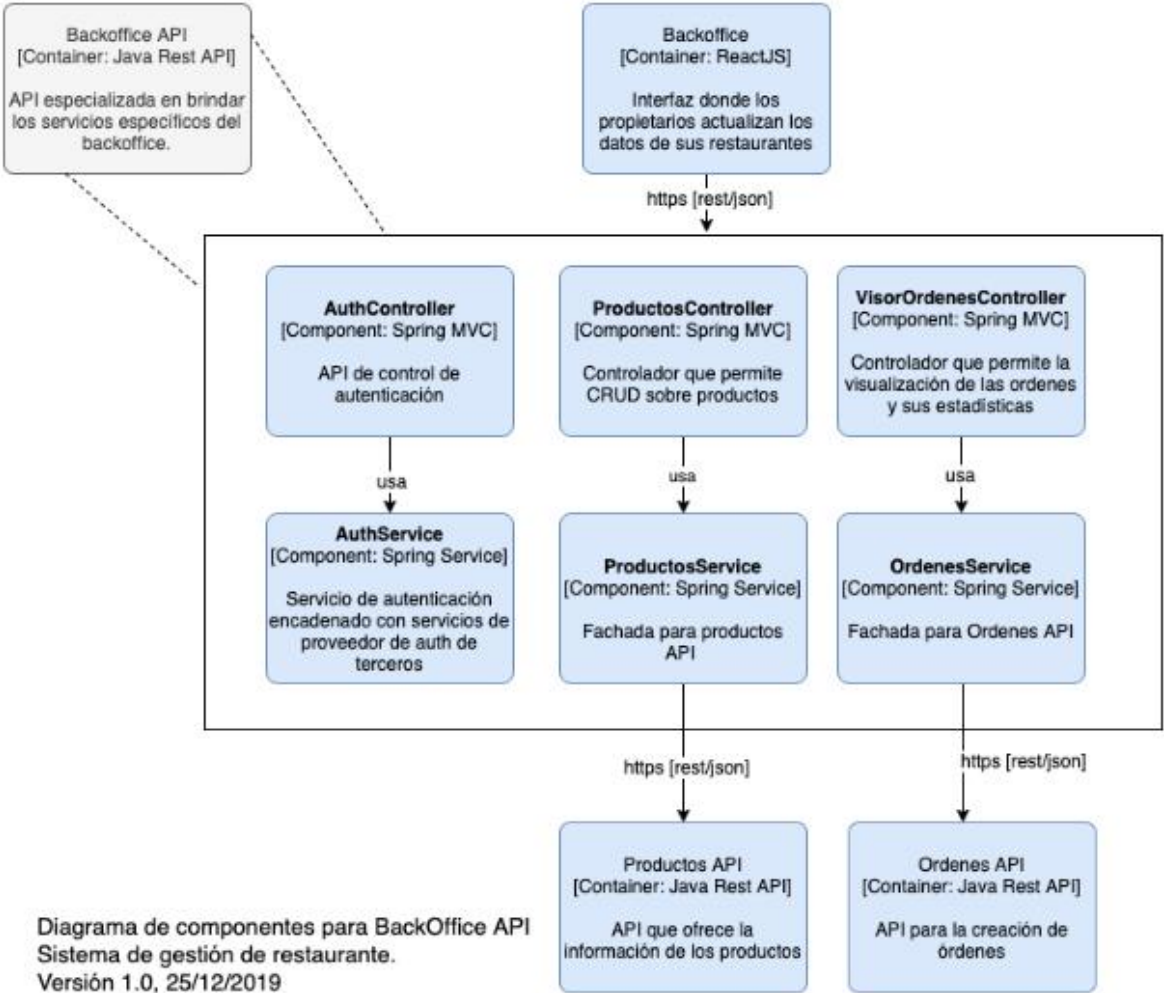
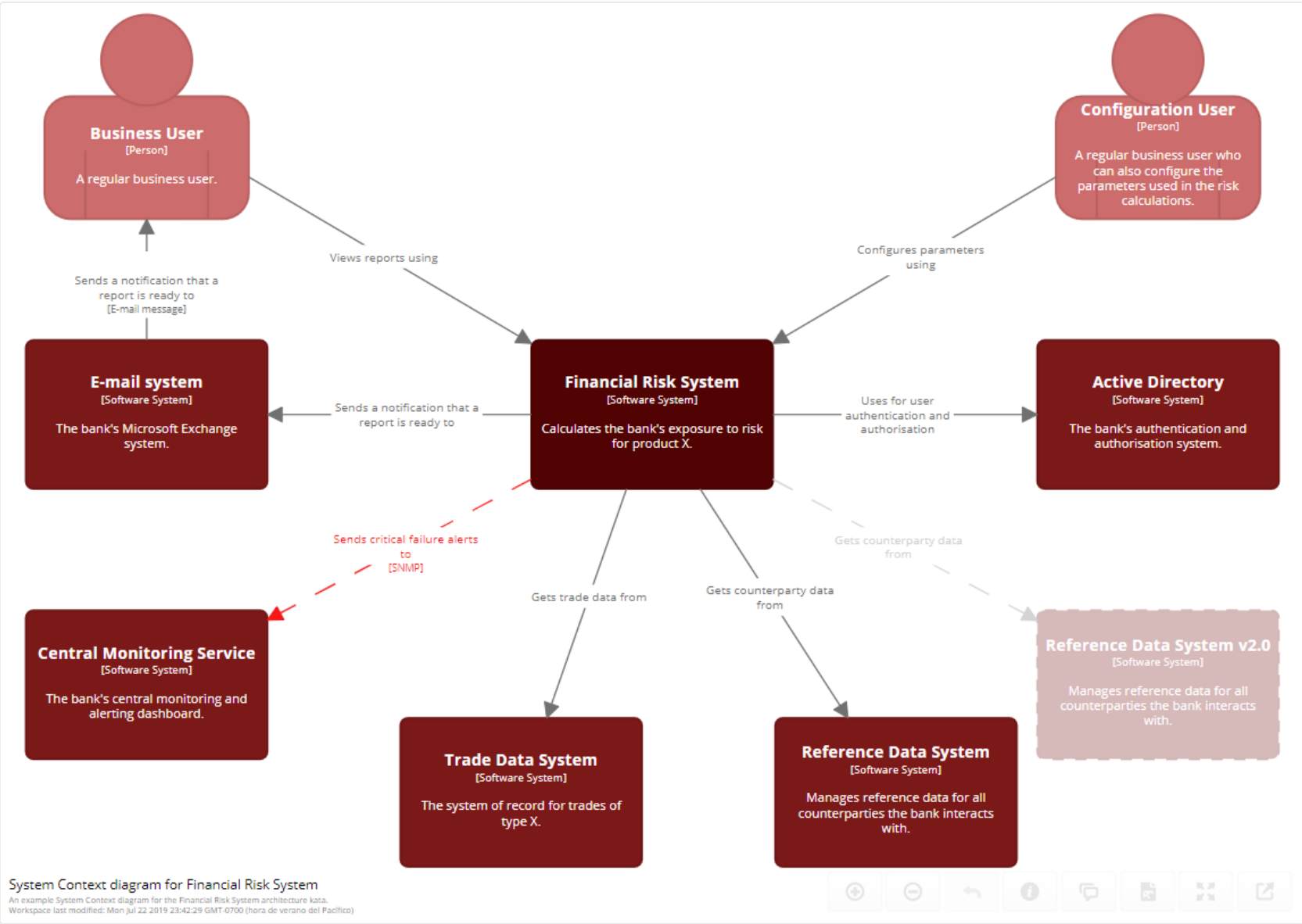


Diagrama de componentes para BackOffice API  
Sistema de gestión de restaurante.  
Versión 1.0, 25/12/2019

Componente BackOffice API

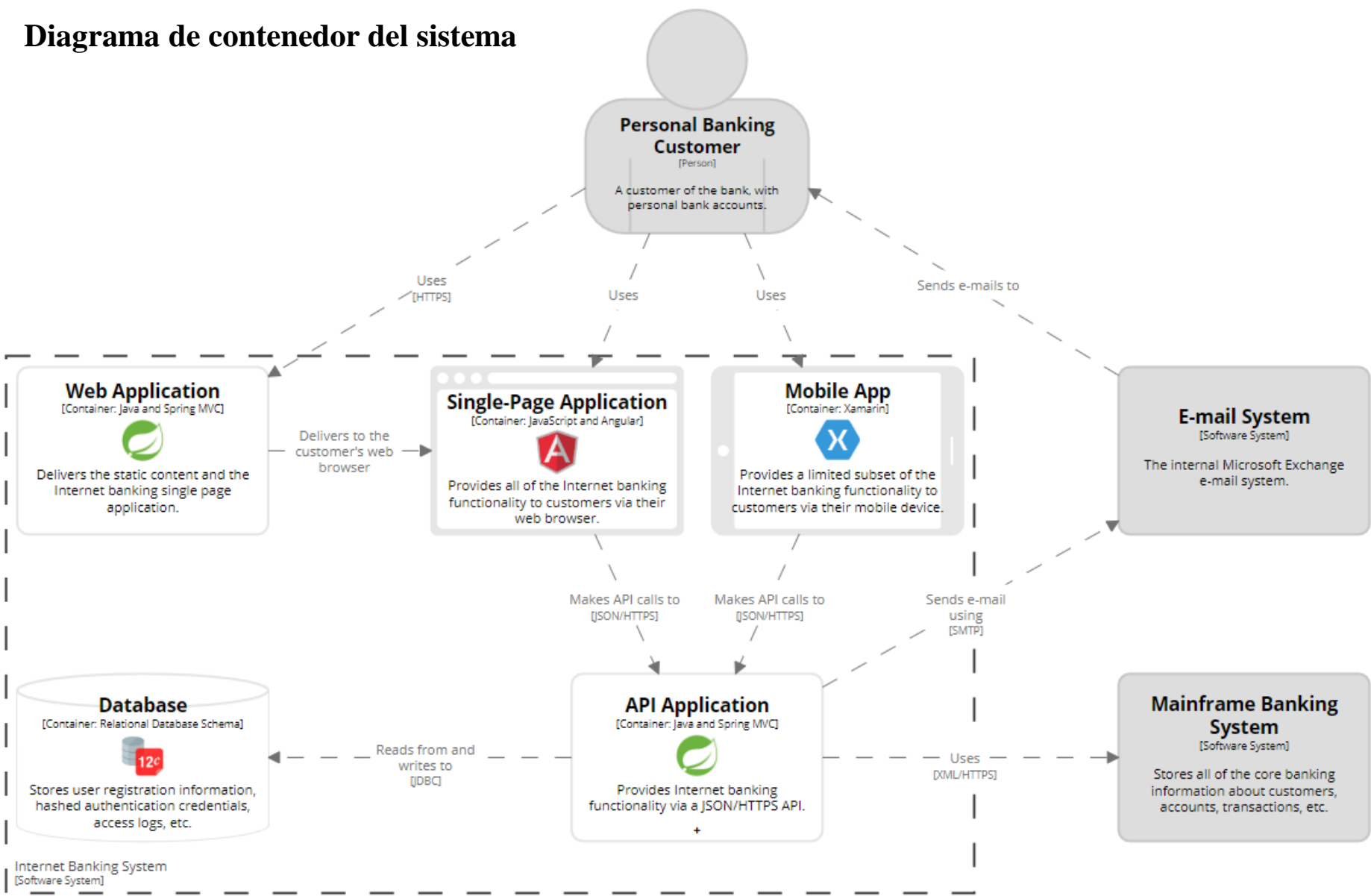
# Modelo C4 Nivel 1 Diagrama de contexto (Ejemplo)

## Diagrama de contexto del sistema



# Modelo C4 Nivel 2 Diagrama de contenedor (Ejemplo)

## Diagrama de contenedor del sistema

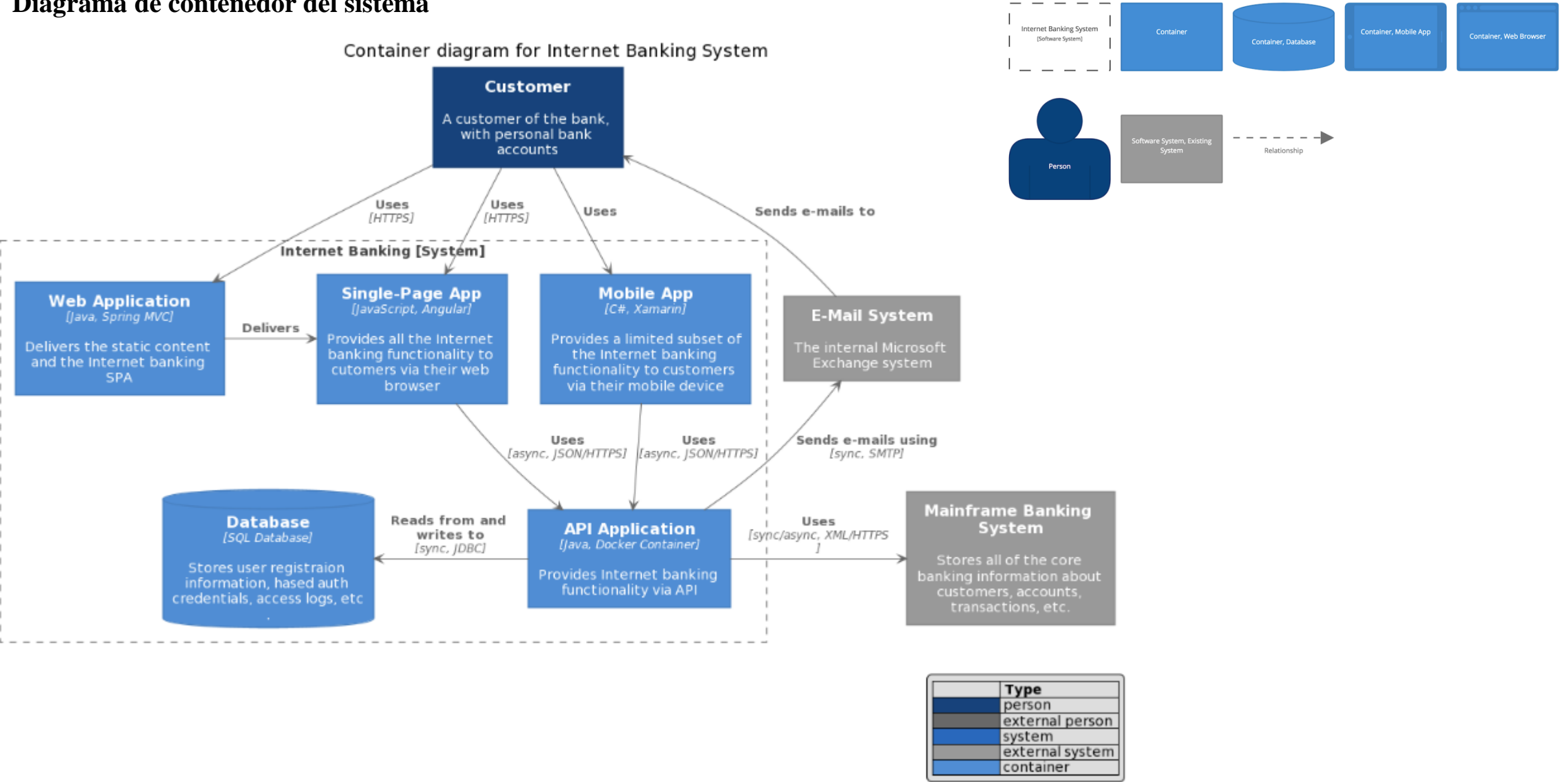


Container diagram for Internet Banking System

The container diagram for the Internet Banking System.  
Workspace last modified: Thu Apr 02 2020 04:02:08 GMT-0700 (hora de verano del Pacífico)

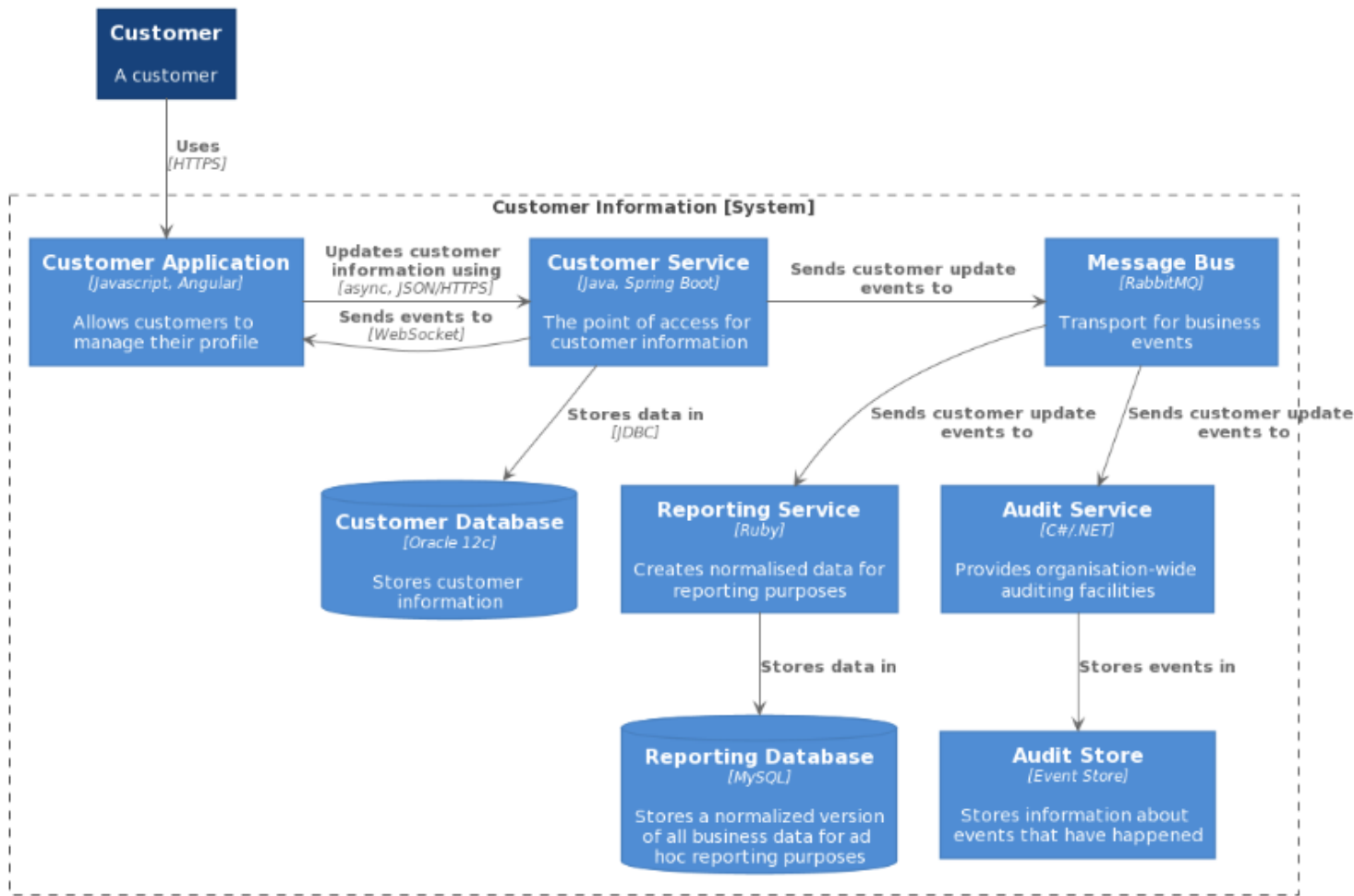
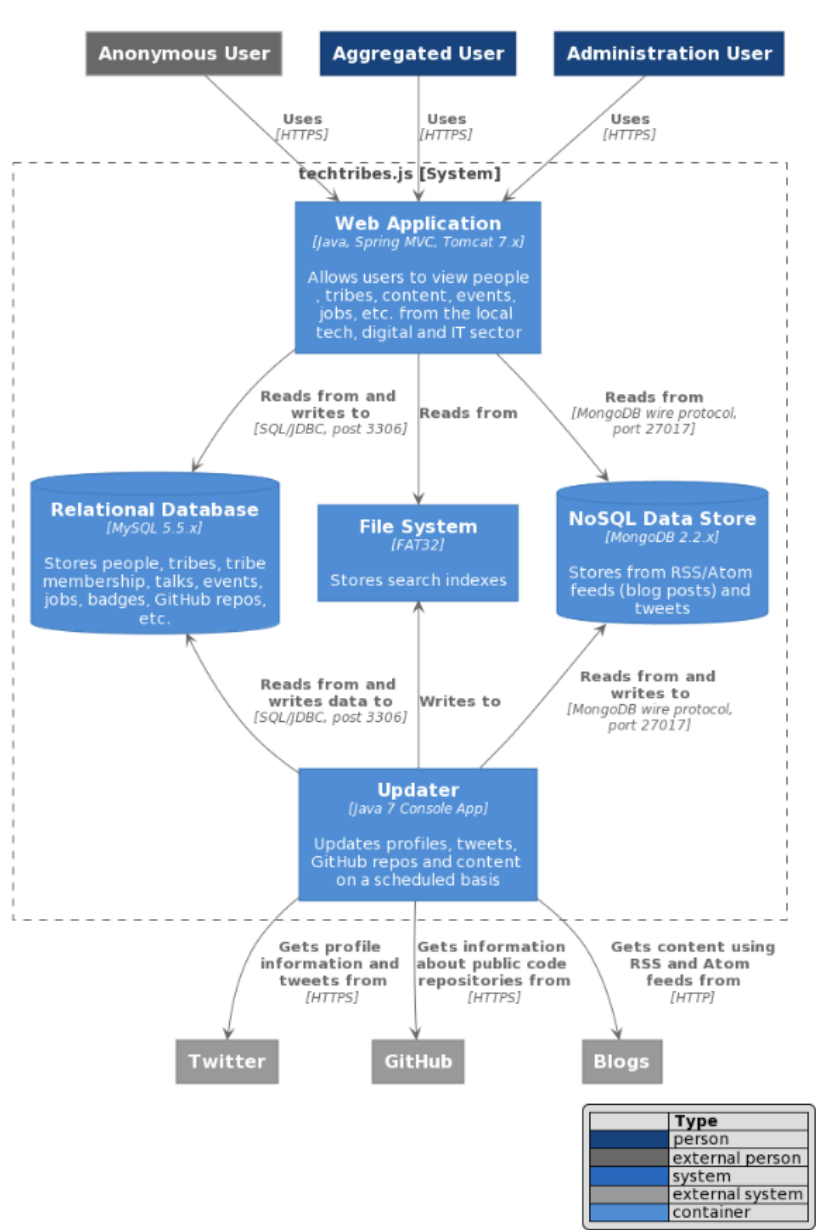
# Modelo C4 Nivel 2 Diagrama de contenedor (Ejemplo)

## Diagrama de contenedor del sistema



# Modelo C4 Nivel 2 Diagrama de contenedor (Ejemplo)

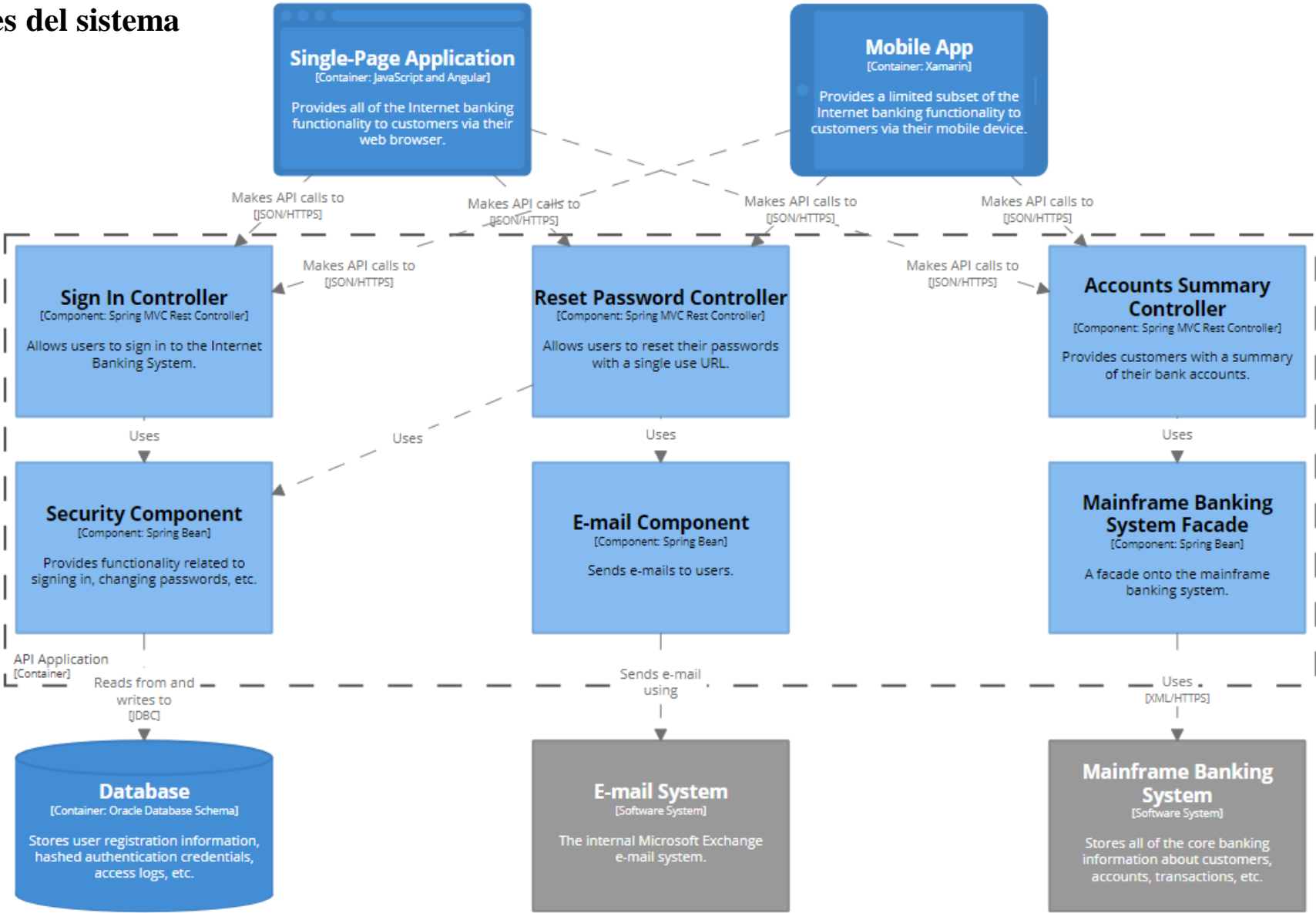
## Diagrama de contenedor del sistema





# Modelo C4 Nivel 3 Diagrama de componentes (Ejemplo)

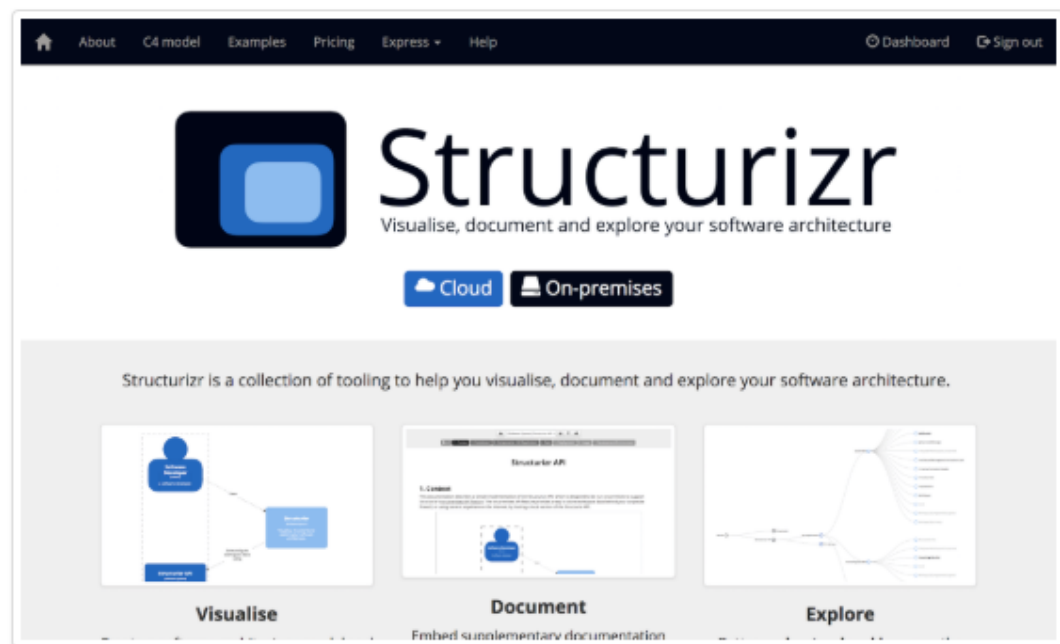
## Diagrama de componentes del sistema



Component diagram for Internet Banking System - API Application

The component diagram for the API Application.  
Workspace last modified: Tue Mar 24 2020 11:29:55 GMT-0700 (hora de verano del Pacífico)

# Modelo C4 Herramientas para el diseño de diagramas



## Structurizr

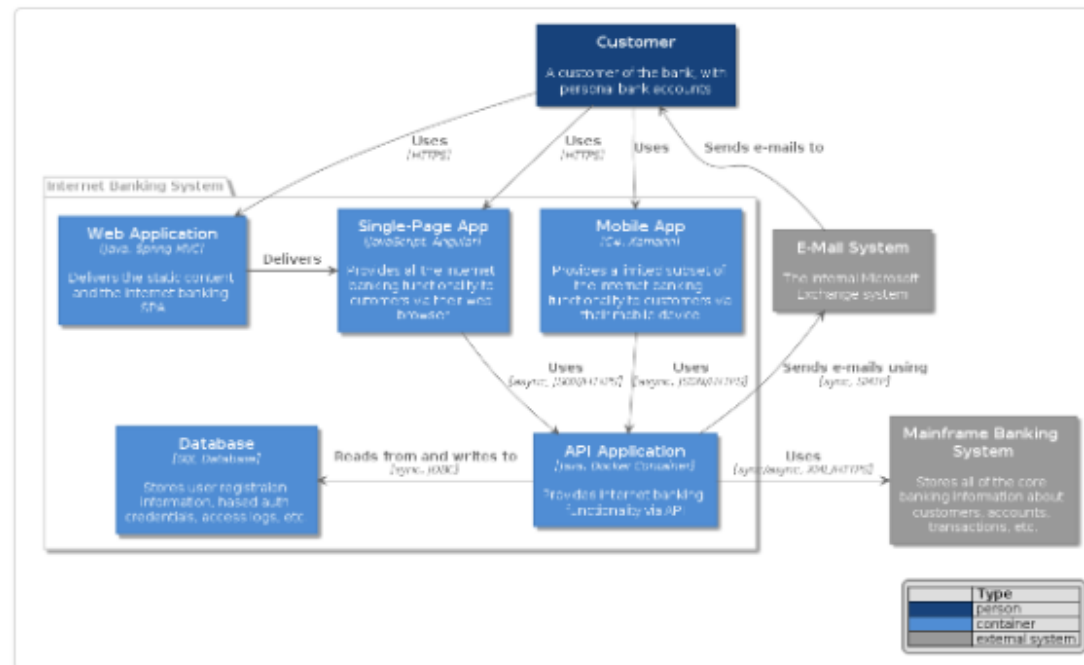
Structurizr is a collection of tooling to help you visualise, document and explore your software architecture. It's an implementation of the C4 model and allows you to create software architecture models using code or a browser-based UI, along with supplementary documentation using Markdown/AsciiDoc.

Client libraries are currently available for [Java](#), [.NET \(core and framework\)](#), [TypeScript](#), and [PHP](#). Alternatively, [arch-as-code](#) is a tool to store software architecture diagrams/documentation as YAML, and publish it to Structurizr.



<https://structurizr.com/>

<https://github.com/structurizr/dotnet>



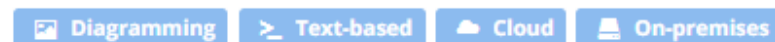
## PlantUML

There are a number of extensions to PlantUML to assist in the creation of C4 model diagrams:

[C4-PlantUML](#) by Ricardo Niepel

[C4-PlantumlSkin](#) by Savvas Kleanthous

[c4builder](#) by Victor Lupu



<https://adrianvlupu.github.io/C4-Builder/#/?id=overview>

<https://github.com/RicardoNiepel/C4-PlantUML>

<https://github.com/skleanthous/C4-PlantumlSkin>

<https://github.com/adrianvlupu/C4-Builder>

# Bibliografía

Pressman, R. S. (2010). Ingeniería de Software, Un enfoque practico Séptima Edición. Ciudad de México: Mc Graw Hill.

Sommerville. (2011). Ingeniería de Software 9 Edición. Estado de México: Pearson.

UNID Universidad Interamericana para el desarrollo. (2018). Ingeniería de software. Ciudad de México: UNID.

## Enlaces

- ✓ <https://c4model.com/>
- ✓ <https://diagrams-as-code.com/#Implementations>
- ✓ <https://github.com/structurizr/dotnet>