

DISEÑO Y PROGRAMACIÓN ORIENTADA A OBJETOS - CREACIÓN DE HISTORIAS DE USUARIO, MODELO DE DIAGRAMAS C4

```
34     self.logout()
35     self.debug = settings.getbool('SUPERVISOR_DEBUG')
36     if path:
37         self.title = path
38     self.file.seek(0)
39     self.fingerprints.update(self.request_fingerprint())
40
41     @classmethod
42     def from_settings(cls, settings):
43         debug = settings.getbool('SUPERVISOR_DEBUG')
44         return cls(job_dir(settings), debug)
45
46     def request_seen(self, request):
47         fp = self.request_fingerprint(request)
48         if fp in self.fingerprints:
49             return True
50         self.fingerprints.add(fp)
51         if self.file:
52             self.file.write(fp + os.linesep)
53
54     def request_fingerprint(self, request):
55         fingerprint(request)
```

CLASE 26

CREACIÓN DE HISTORIAS DE USUARIO

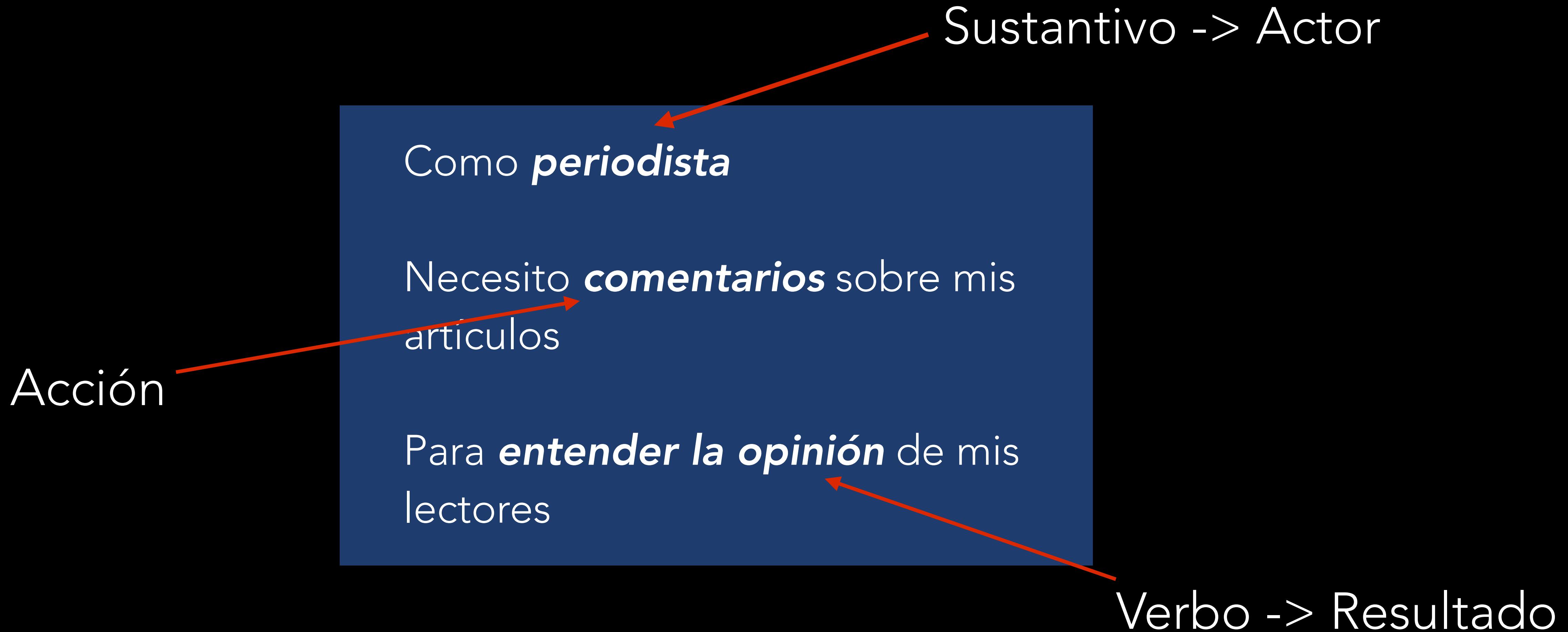
LA IMPORTANCIA DE LA DEFINICIÓN DE OBJETIVOS CLAROS

- Un set de objetivos claros son el puntapié inicial para poder definir metas logrables y medibles.
- Debemos considerar que una historia de usuario debería poder ser realizada dentro de una iteración, por tanto, debe ser acotada de tal forma que el objetivo y los criterios de aceptación permitan ello.

IDENTIFICACIÓN DE LOS PRINCIPALES INVOLUCRADOS

- Para que una historia tenga asidero, es necesario identificar quienes serán los actores involucrados en esta historia, a quienes afectará o beneficiará el desarrollo de la historia.

FORMATO DE LA IDENTIFICACIÓN DE LOS INVOLUCRADOS (CARD)



CREACIÓN DE UNA HISTORIA DE USUARIO

- En el refinamiento el equipo toma nota de lo que se vaya ratificando, siendo la piedra angular que permitirá definir eventuales antecedentes de entrada (Conversation)

CREACIÓN DE UNA HISTORIA DE USUARIO

- Se definen criterios de aceptación de la historia, los cuales pueden definir:
 - Valores esperados
 - Límites posibles
 - Cambios de estado
 - Entre otros...

CREACIÓN DE UNA HISTORIA DE USUARIO

- Estos mismos criterios de usuario se pueden entender como los casos de uso que nuestros tests unitarios (y de integración) deberían abordar para garantizar el cumplimiento de la historia.

¿CÓMO PRESENTAMOS NUESTRO DISEÑO DE SOFTWARE?

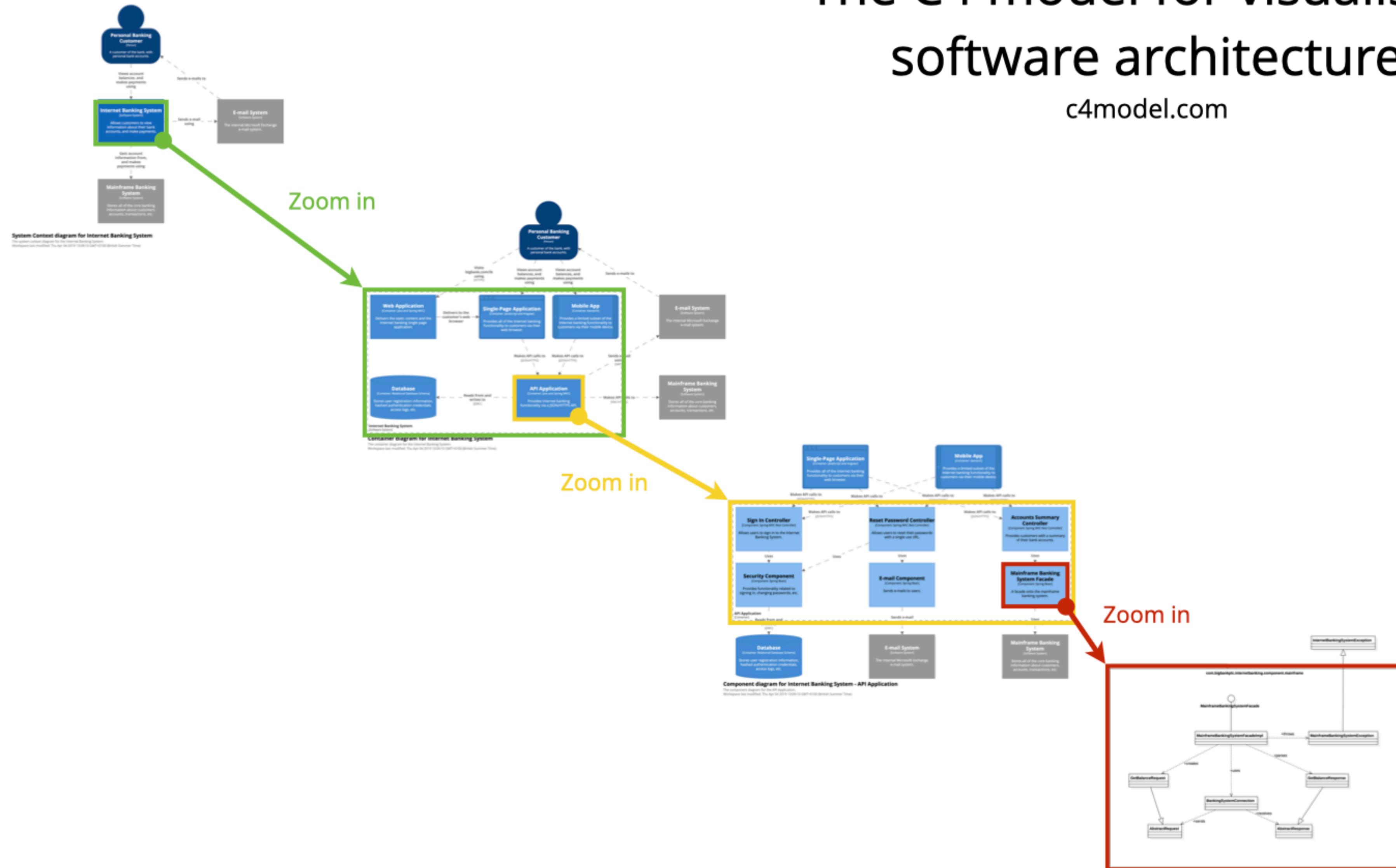
DIAGRAMAS C4

DIAGRAMAS C4

- El modelo de diagramas c4 permite visualizar el diseño de software en todos sus niveles, considerando desde la arquitectura hasta el diseño de clases.
- En un nivel más detallado, finalmente se transforma en un diagrama de clases.
- Permite dar un entendimiento de alto nivel, a todas las áreas involucradas, las que no necesariamente pueden ser técnicas.

The C4 model for visualising software architecture

c4model.com



Level 1
Context

Level 2
Containers

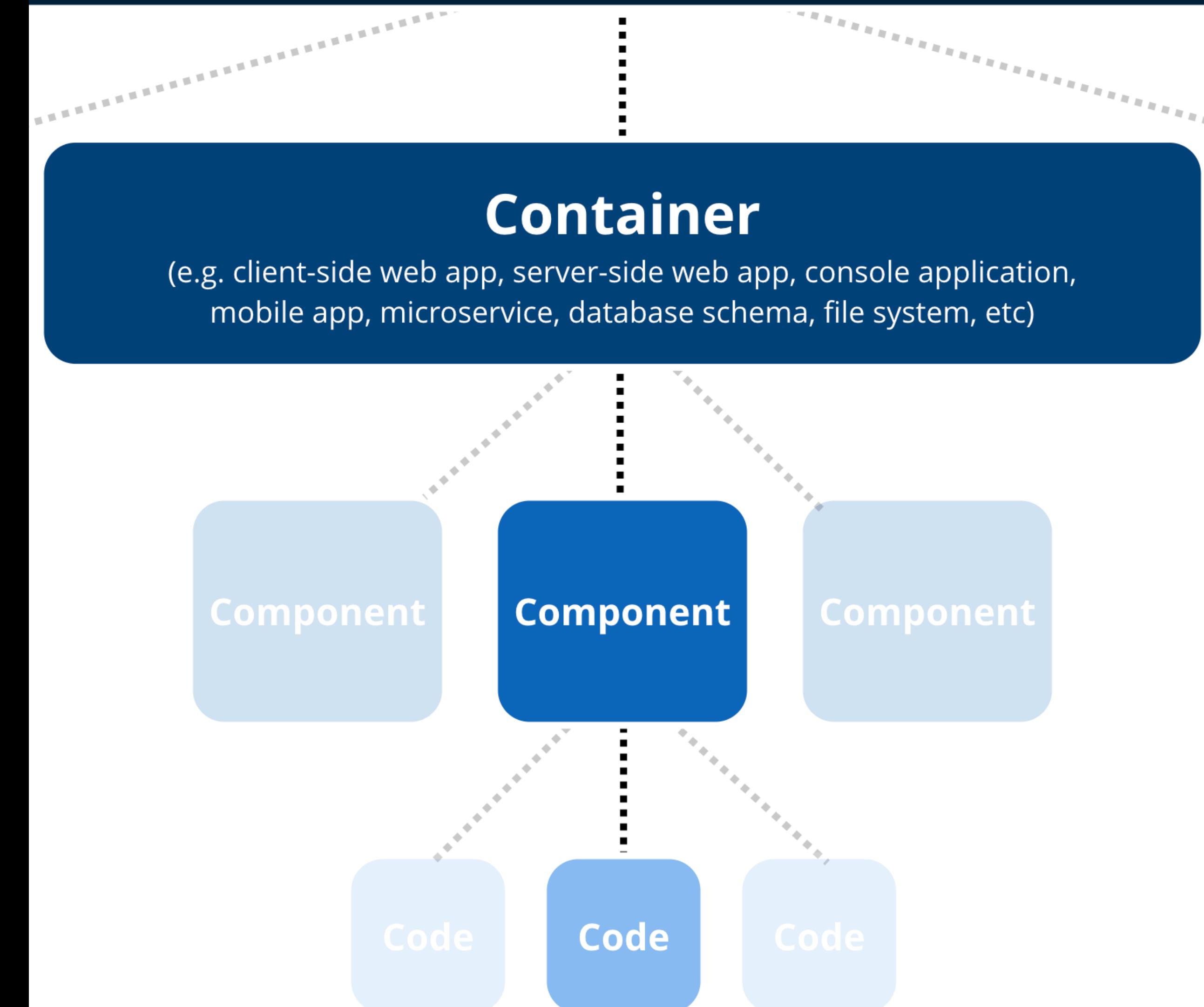
Level 3
Components

Level 4
Code

ABSTRACCIONES

- Con el fin de crear mapas de nuestro código, es necesario contar con una abstracción que permita describir la estructura de un sistema de software.
- El modelo c4 considera las estructuras estáticas de un sistema de software en términos de contenedores, componentes y código. Y finalmente, la gente que usa el software que construimos.

Software System

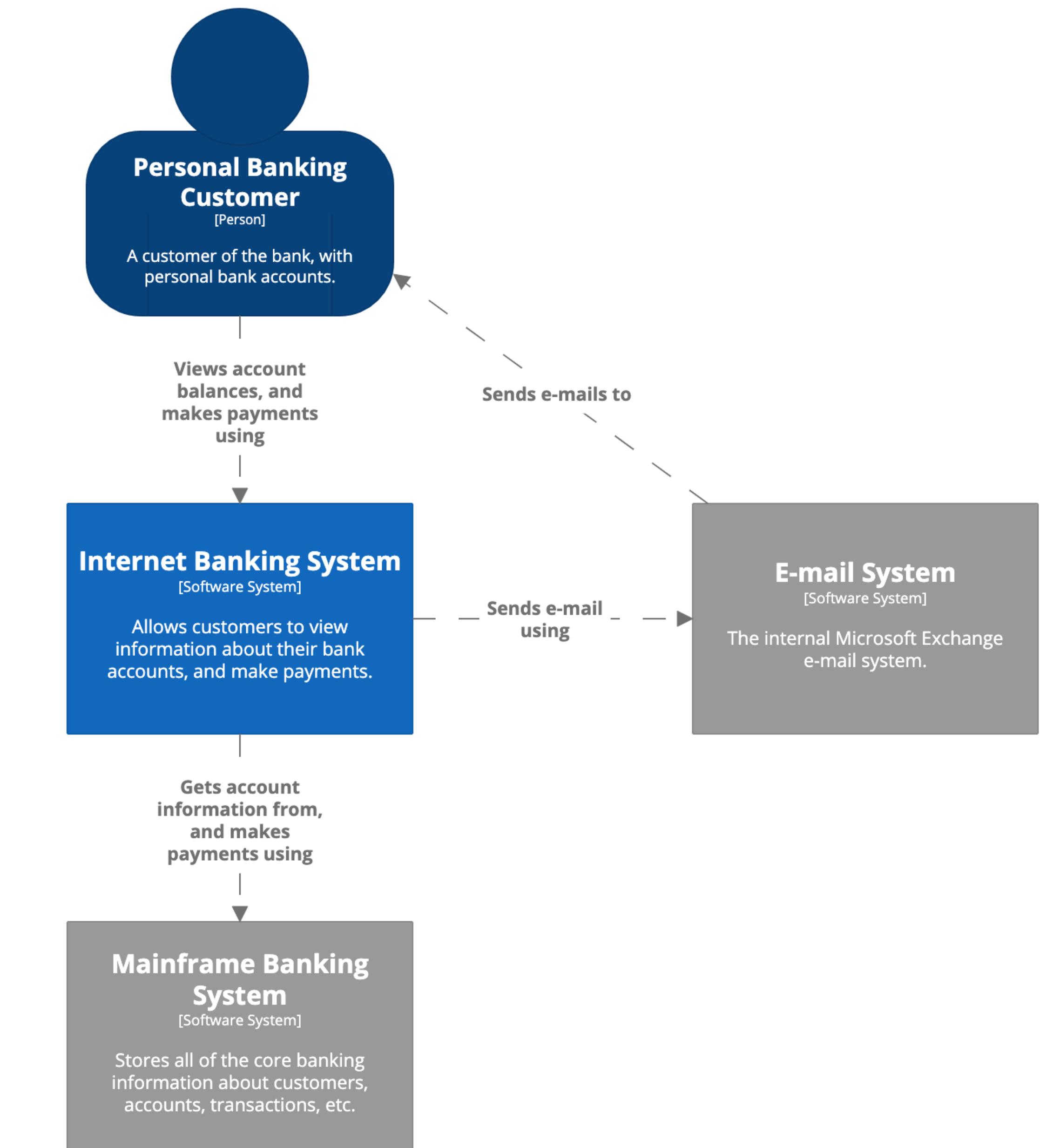


ABSTRACCIONES

- Persona: Representa a un humano que usa nuestro sistema de software
- Sistema de software: Es el mayor nivel de abstracción y describe aquello que entrega valor a los usuarios.
- Contenedor: No es docker!, un contenedor representa a una aplicación o almacén de datos. Corresponde a un artefacto que necesita correr en conjunto con otros para que el sistema funcione.
- Componentes: Corresponde a un grupo de funcionalidades encapsuladas detrás de una interfaz bien definida. Si consideramos el uso de C++ o Java, los componentes corresponderían a la colección de implementación de clases.

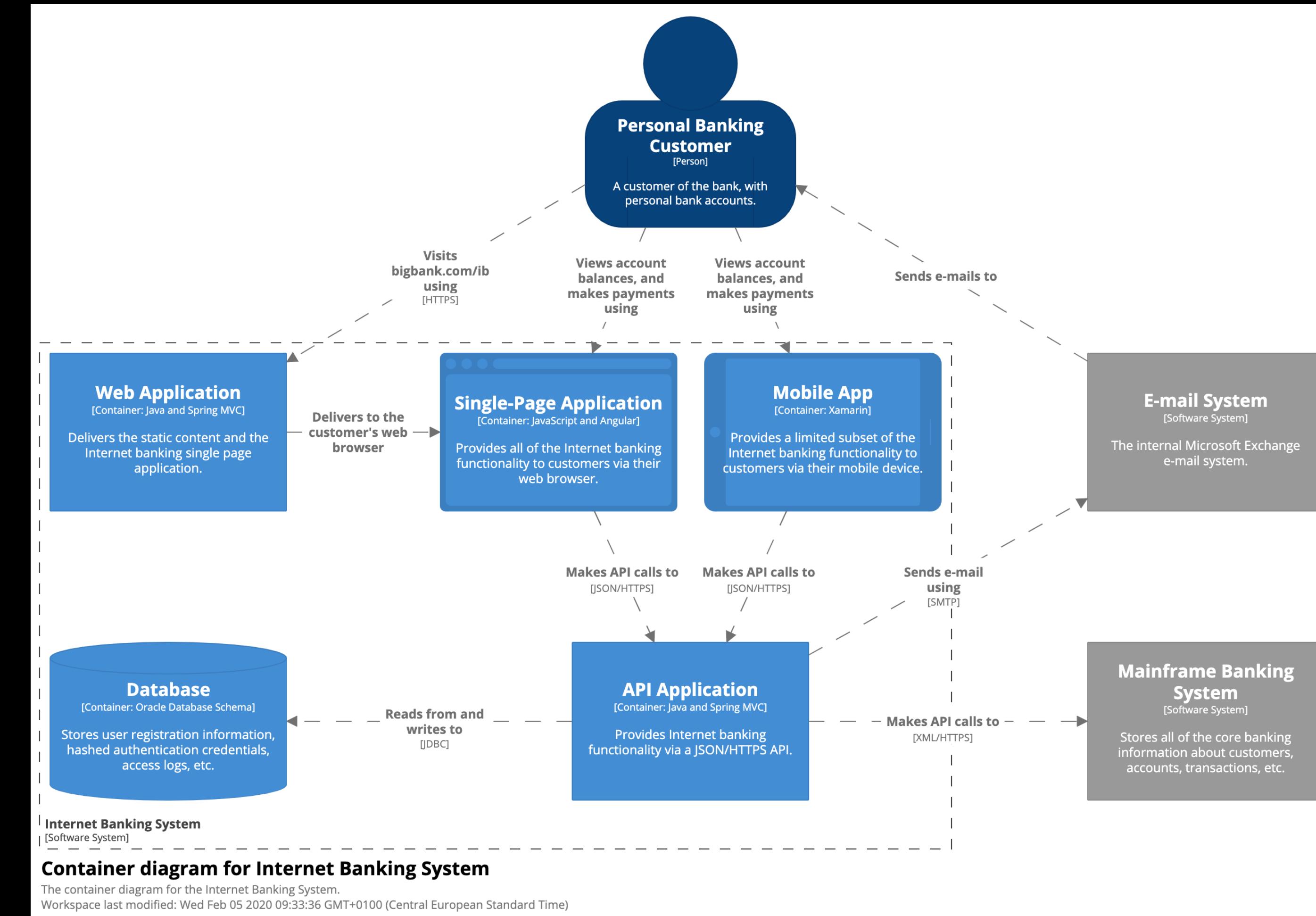
DIAGRAMAS

- Nivel 1: Diagrama de contexto de sistema
- Permite ver el alto nivel, quienes interactúan con nuestro sistema



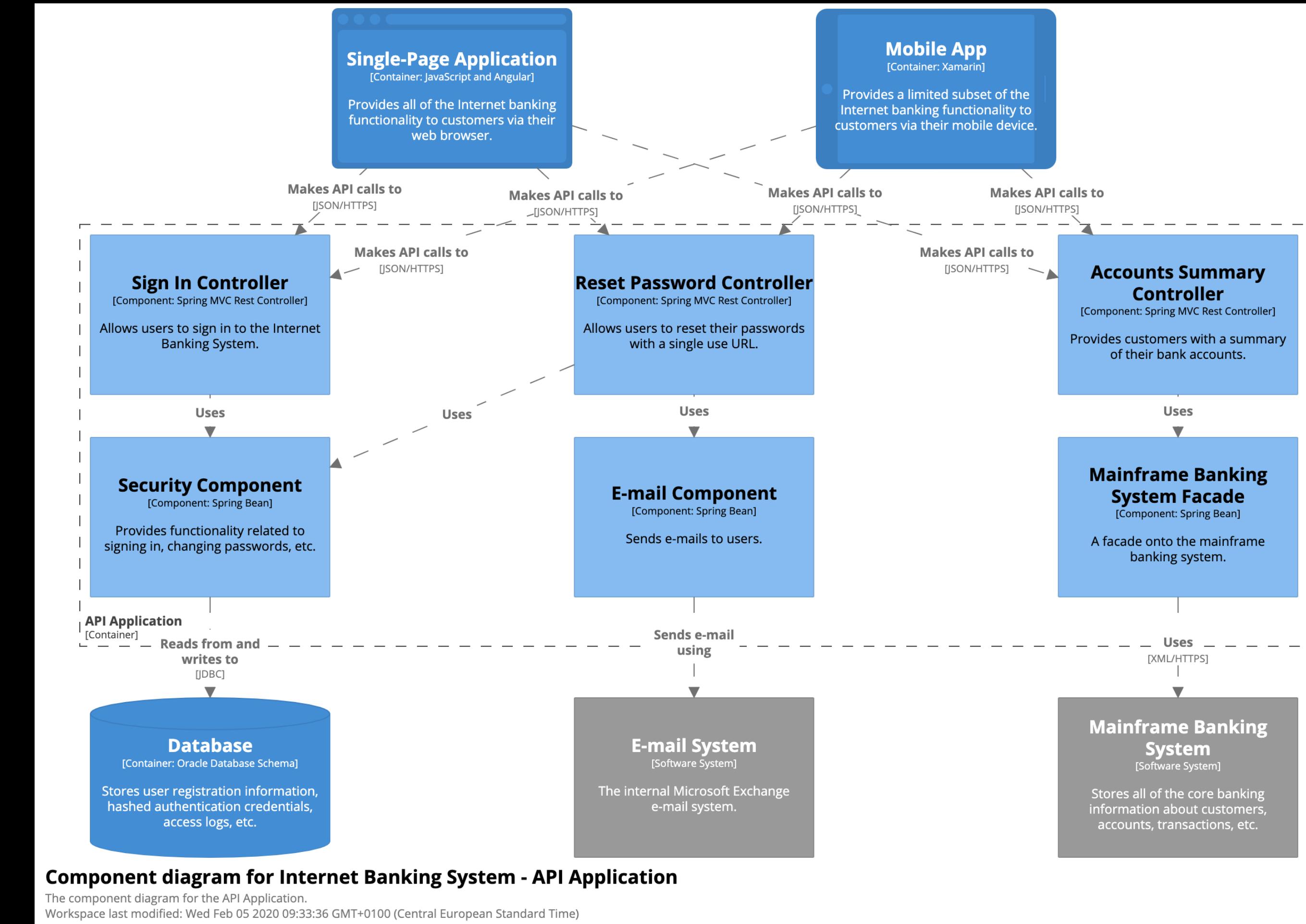
NIVEL 2: DIAGRAMA DE CONTENEDORES

- Una vez que se entienda cómo el sistema interactúa, es necesario hacer doble click para entender qué aplicaciones son las que interactúan entre sí.
- Permite ver más detalle de las interacciones.



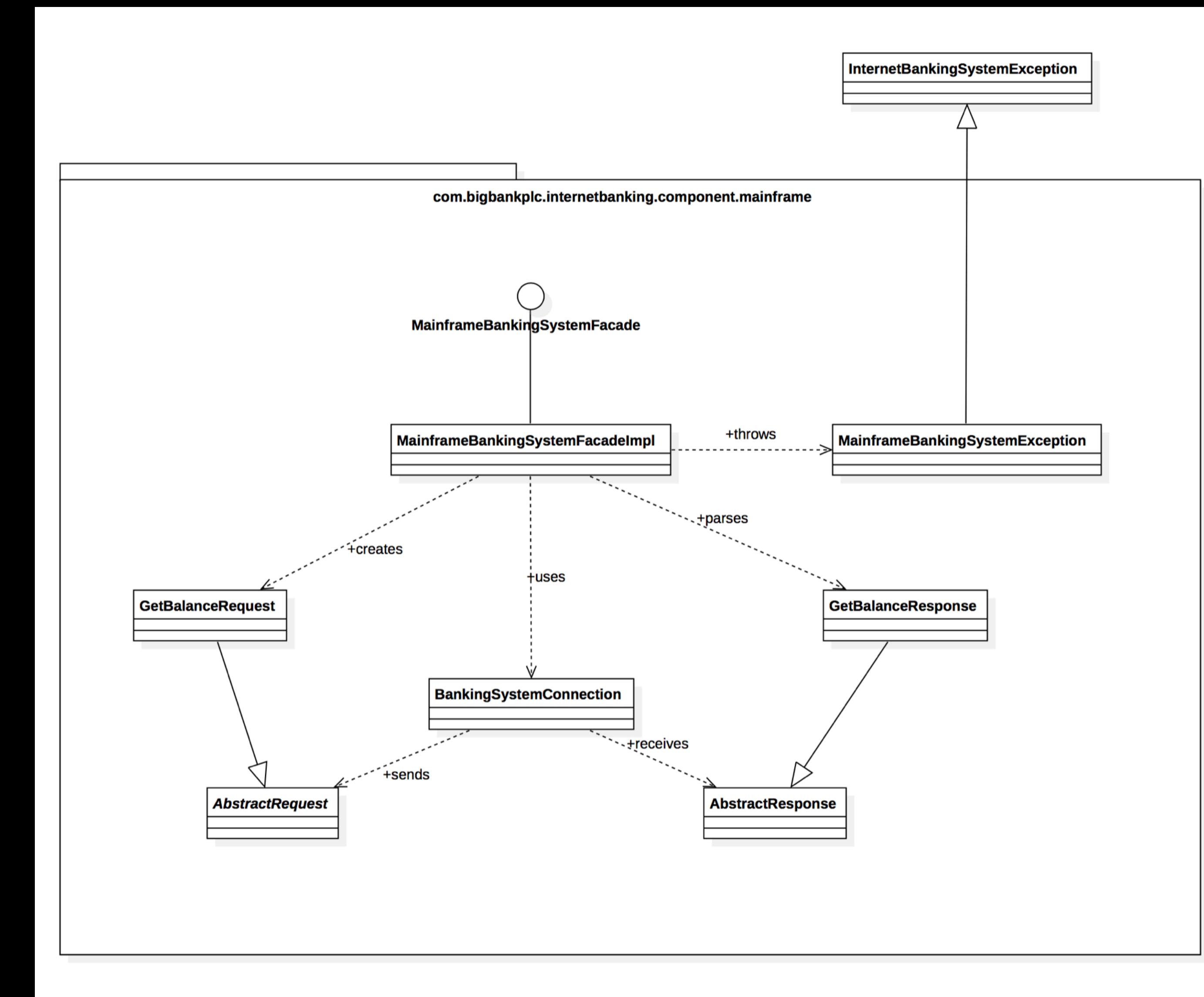
NIVEL 3: DIAGRAMA DE COMPONENTES

- Al hacer nuevamente doble click, podemos visualizar los componentes más finos que interactúan entre sí, es decir, controladores, bases de datos, etc.



NIVEL 4: CÓDIGO

- En este nivel, llegamos a los ya conocidos diagramas UML, diagramas de clase, entidad-relación o similares.
- Nos permiten llegar al nivel de detalle necesario para entender el acoplamiento entre las clases e interfaces.



OTROS DIAGRAMAS...

- Existen otros diagramas complementarios:
 - Diagrama dinámico
 - Diagrama de deployment
 - Entre otros...

NOTACIÓN



- Es posible utilizar herramientas para generar nuestros propios diagramas:
 - PlantUML
 - diagrams.net
 - OmniGraffle
 - Entre otros...

Más información: <https://c4model.com/>

FIN CLASE 26

GRACIAS POR SU ASISTENCIA
¿TIENEN PREGUNTAS?