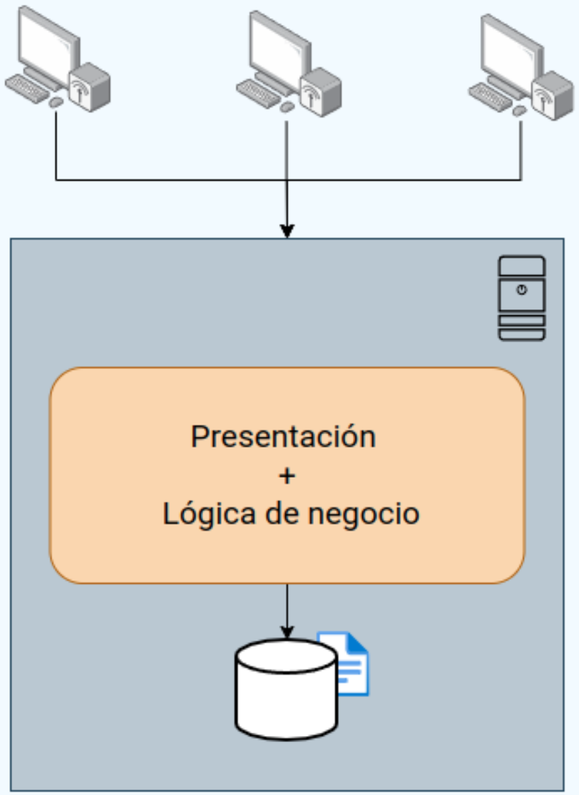
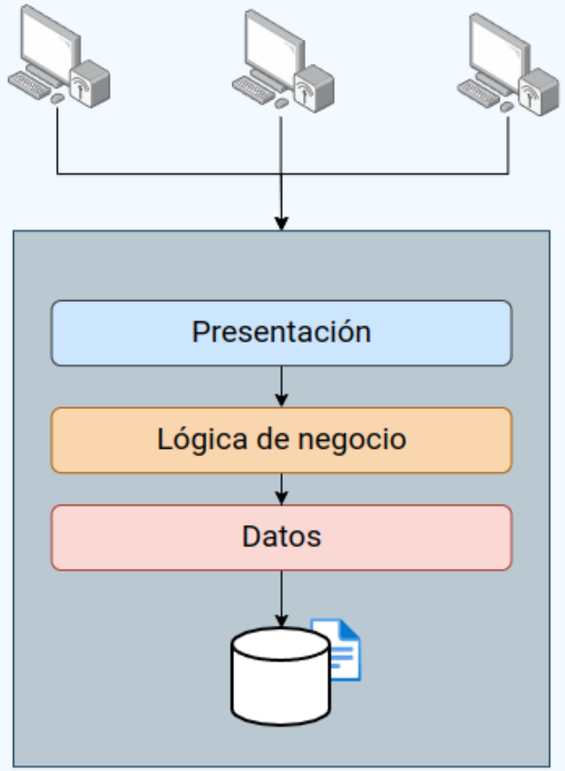
DOMAIN DRIVEN DESIGN

ARQUITECTURA DE CAPAS

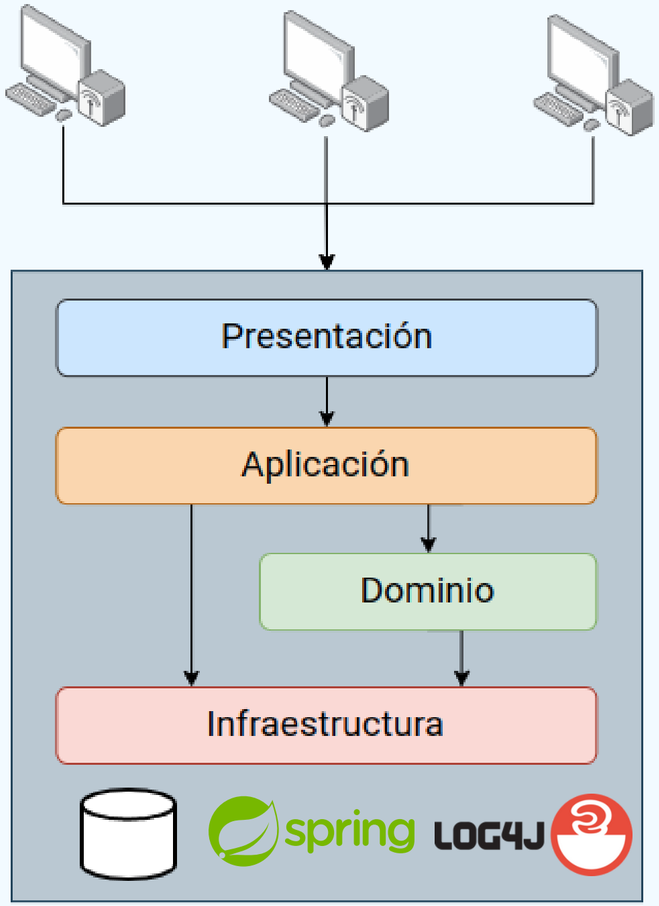
Daniel Blanco Calviño

ARQUITECTURA DE 3 CAPAS

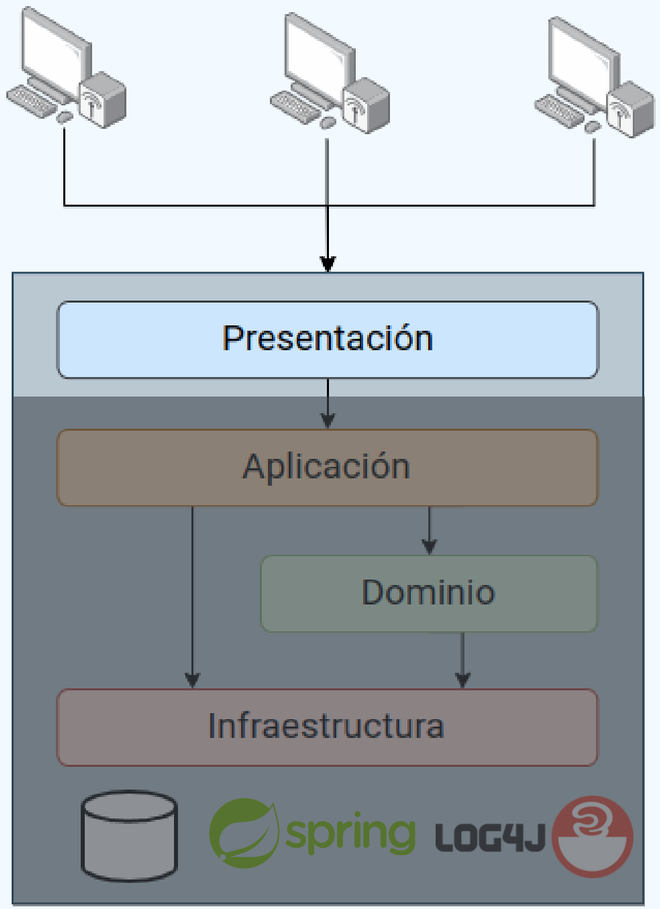




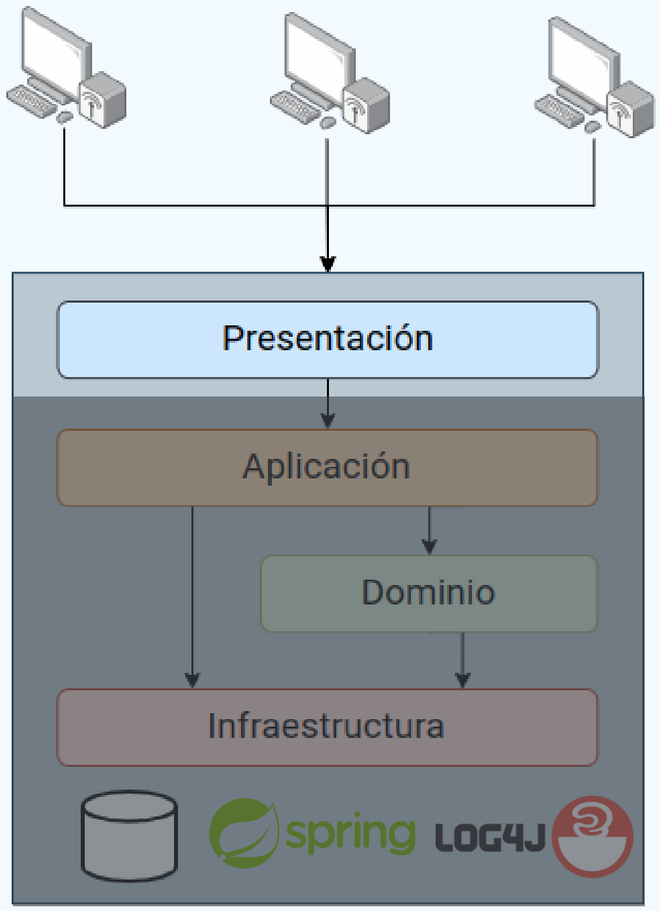
ARQUITECTURA DE CAPAS DDD



CAPA DE PRESENTACIÓN



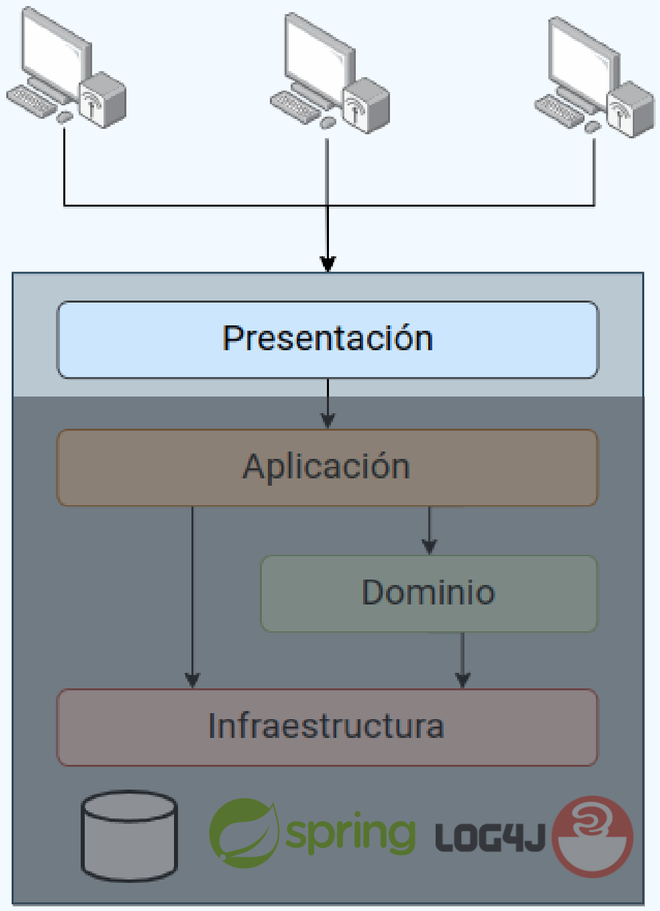
CAPA DE PRESENTACIÓN



Interfaz de usuario

- En el pasado, se construía la vista desde el servidor, por eso **se consideraba parte de la presentación.**
- Ahora es muy común tener la vista separada. React, Angular, Vue etc.

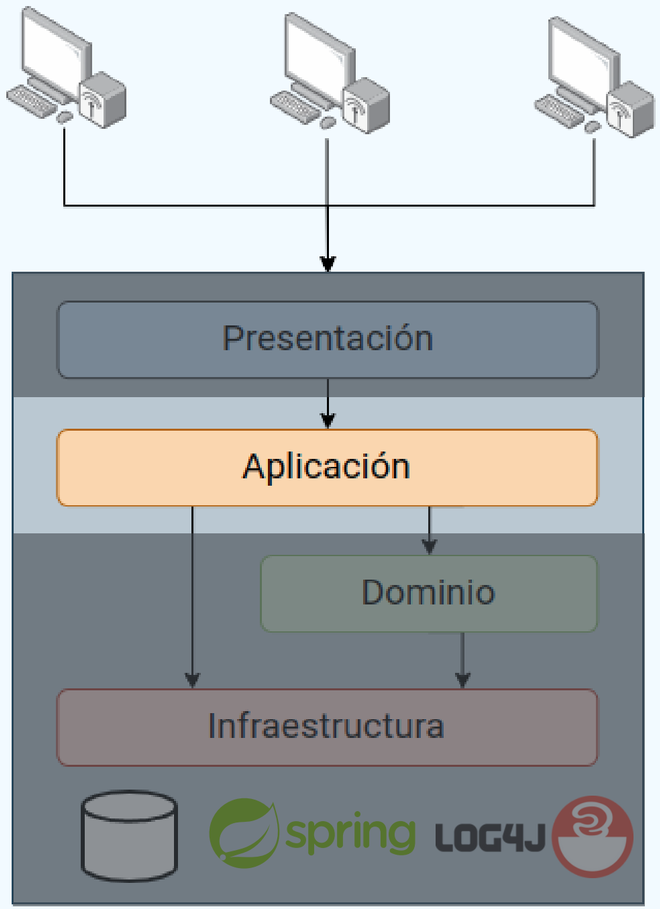
CAPA DE PRESENTACIÓN



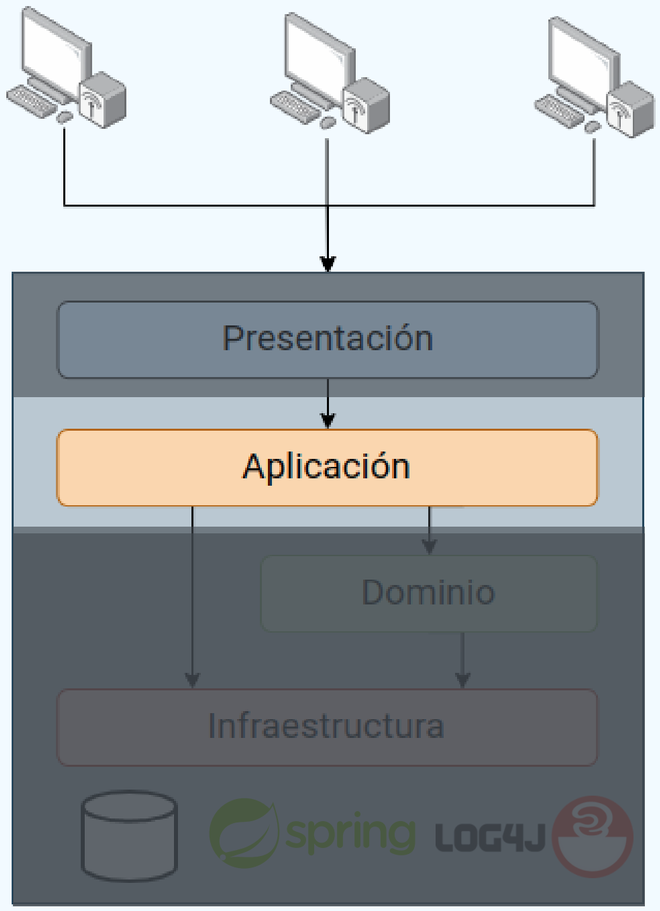
- API de entrada a nuestro sistema que da soporte a la interfaz de usuario.
- Es la fachada e interactúa con los servicios de aplicación para iniciar los casos de uso.

@Controller en Spring

CAPA DE APLICACIÓN

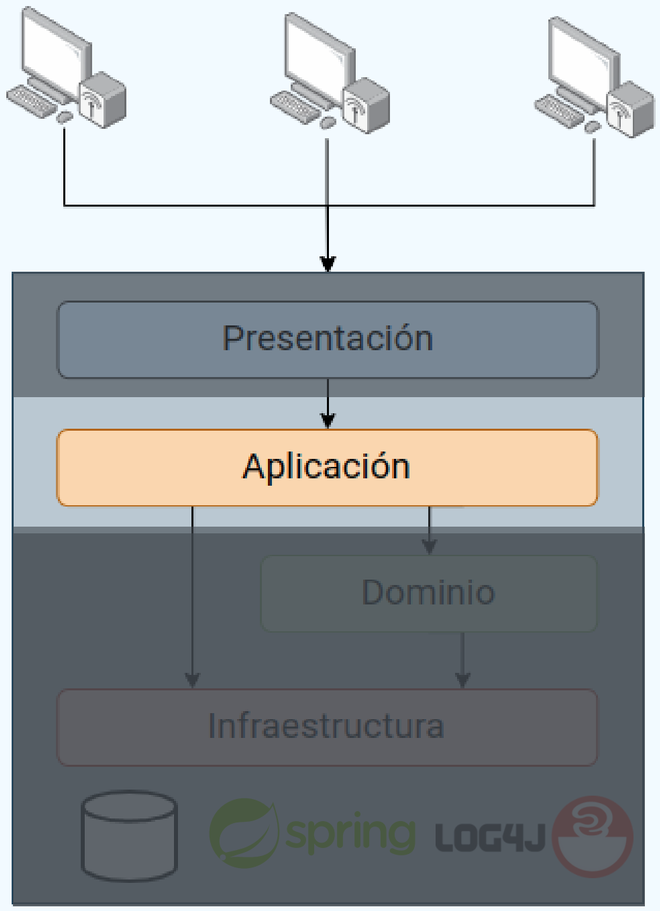


CAPA DE APLICACIÓN



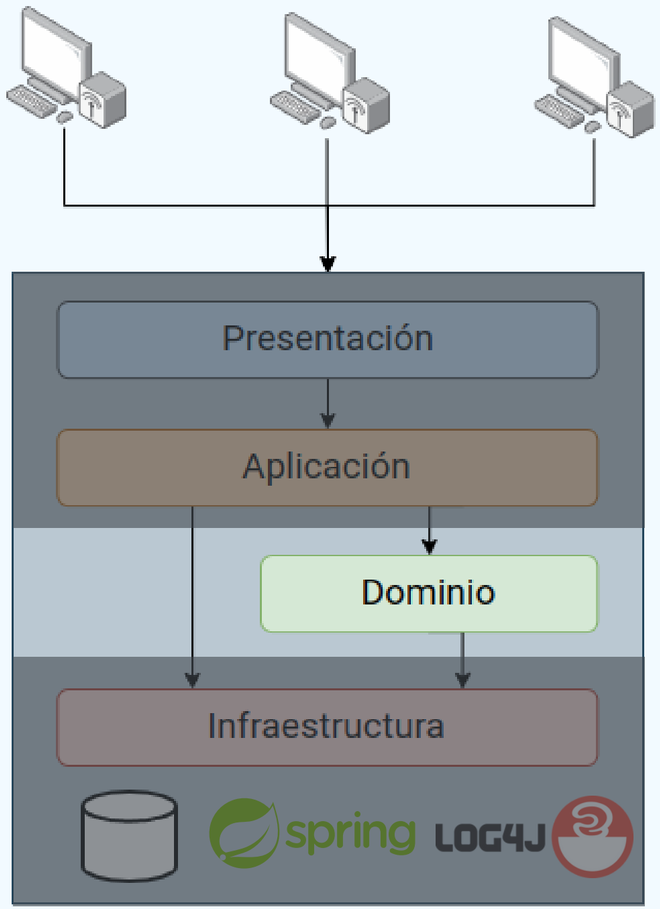
- Encargada de **orquestar** todos los **casos de uso** necesarios para el funcionamiento de nuestro sistema.
- **Interactúa con el dominio** para ejecutar su lógica específica.
- **Interactúa con la infraestructura** para la persistencia, framework, logging etc.
- **Responde a la presentación** con los datos formateados correctamente.

CAPA DE APLICACIÓN

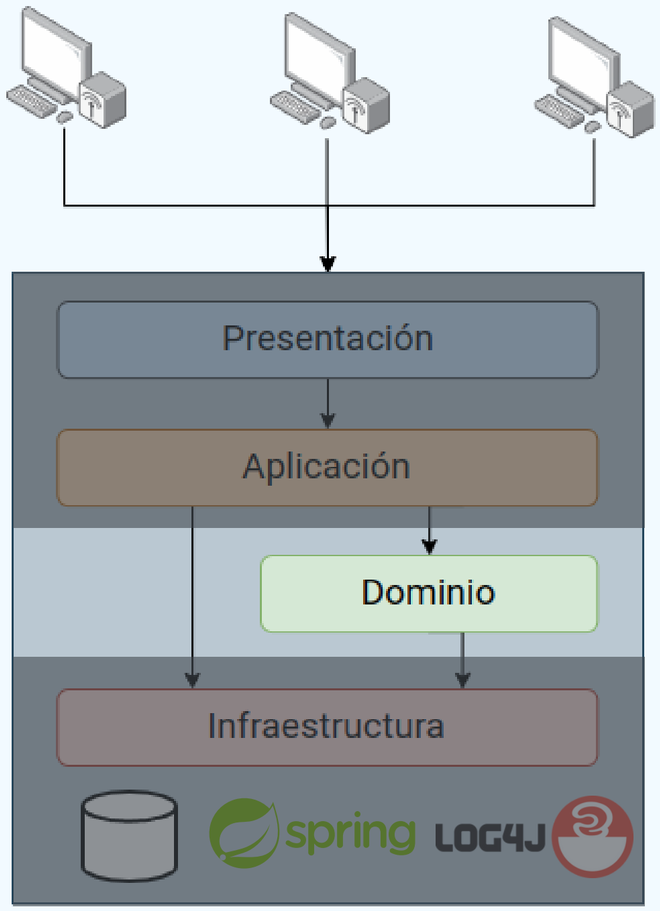


@Service en Spring

CAPA DE DOMINIO

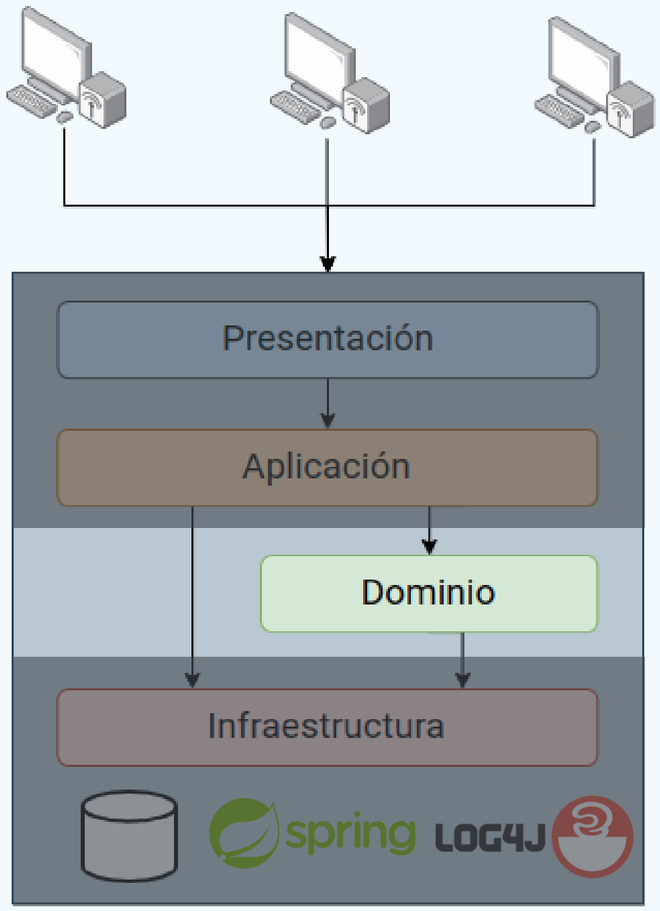


CAPA DE DOMINIO



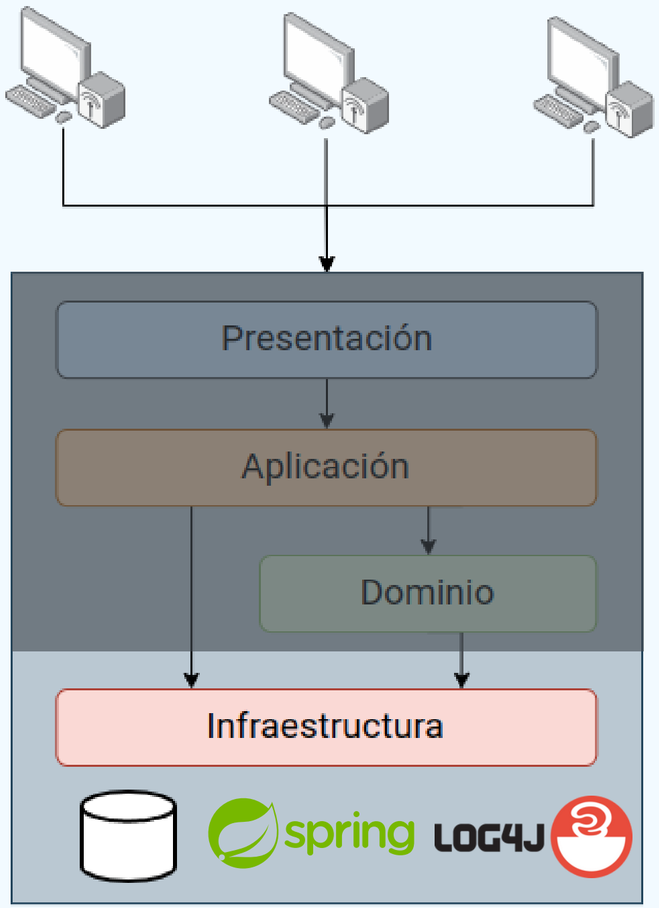
- **Datos y lógica central de nuestro sistema**, diseñada bajo los principios de DDD.
- Debe estar **lo más aislado posible del exterior**. Se comunica con infraestructura si necesita algún aspecto como logging.
- Se compone de entidades de dominio y servicios de dominio.

CAPA DE DOMINIO



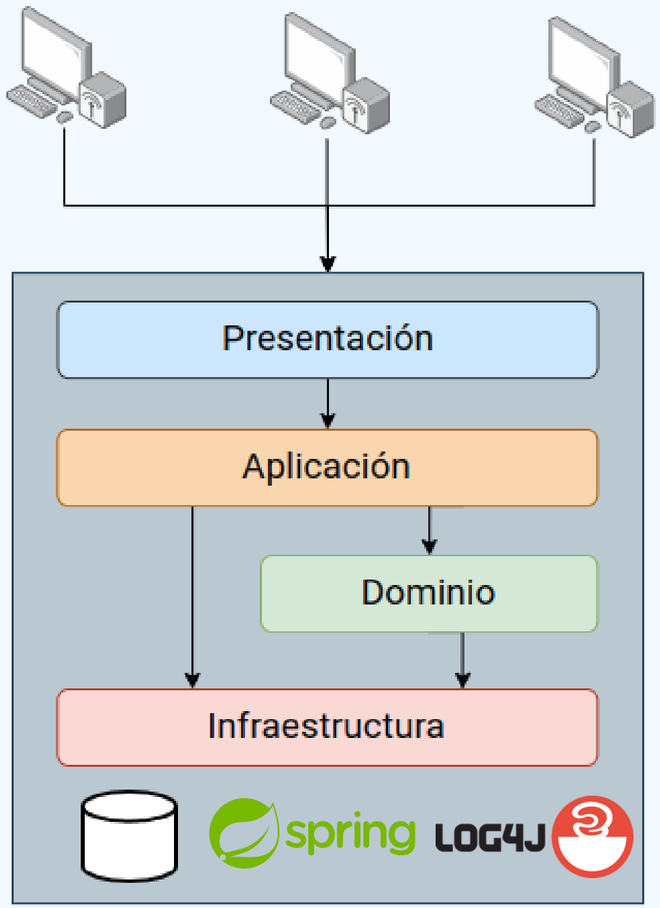
- **Entidades de dominio**
  - Datos.
  - Lógica.
  - **NO** son entidades de persistencia (**@Entity** en Spring).
- **Servicios de dominio**
  - Lógica de dominio que no se pueda asignar a una entidad de dominio específica.
  - Siguen los principios del DDD.
  - **@Service** en Spring.

CAPA DE INFRAESTRUCTURA



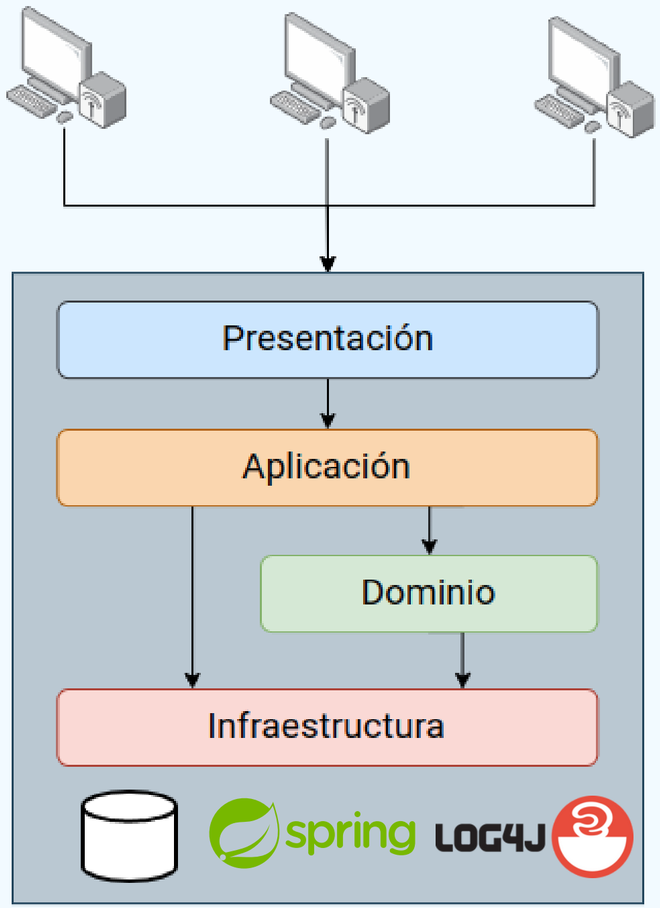
- **Persistencia.**
  - Objetos de ORMs (@Entity etc).
  - Repositorios.
- **Detalles del Framework**
  - Clases de configuración.
  - Arranque de la aplicación.
- **Otros aspectos de infraestructura**
  - Logging.

DEPENDENCIAS



- **Las flechas indican el flujo de la información**, no la dependencia.
- Es importante conseguir la **IoC** usando técnicas como la **inyección de dependencias.**

DEPENDENCIAS



- Es importante que el **dominio sea lo más estable** de nuestro sistema.
- Nunca debemos modificar nuestro dominio para adaptarlo al exterior, como por ejemplo BBDD.
- En ese caso, lo apropiado es adaptar la capa de infraestructura.

DOMAIN DRIVEN DESIGN

ARQUITECTURA DE CAPAS

Daniel Blanco Calviño