

Desarrollo de software basado en microservicios: un caso de estudio para evaluar sus ventajas e inconvenientes



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Escola Tècnica
Superior d'Enginyeria
Informàtica



etsinf

Autor: Víctor Alberto Iranzo Jiménez

Tutores: Patricio Orlando Letelier Torres

María Carmen Penadés Gramage

Jueves, 20 de Septiembre de 2018

Índice

1. Motivación y objetivos
2. Proceso de desarrollo
3. Estado del arte
4. Caso de estudio
5. Evaluación
6. Conclusiones y trabajo futuro

Arquitectura de microservicios

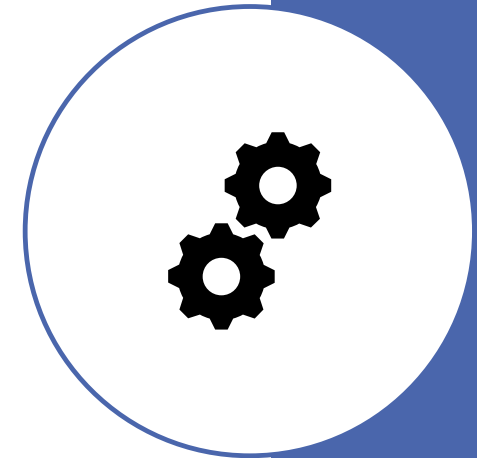
- Servicios: funcionalidades que se exponen a las aplicaciones clientes. Se ejecutan en procesos independientes
- Pequeños: no debe ser el foco principal. Prevalece respetar los principios de alta cohesión y bajo acoplamiento
- Autónomos: evolucionan de forma independiente

Arquitectura monolítica

- Sus módulos no pueden ejecutarse de forma independiente
- ↑ Tamaño ↔ ↑ Complejidad
- Escala como un conjunto

Motivación

- Arquitectura que se adapte a las **necesidades** del negocio
- Profundizar en el conocimiento de las tecnologías de microservicios
- Tener una experiencia de primera mano



Objetivos

- **Desarrollar** una misma aplicación siguiendo dos arquitecturas diferentes: una basada en microservicios y otra monolítica
- Comparar el **proceso de desarrollo** de ambos sistemas
- Evaluar diferentes situaciones durante el **mantenimiento**
- Examinar ambas arquitecturas respecto a diferentes **RNFs**





Proceso de desarrollo

1. Especificación
2. Diseño
3. Implementación
4. Pruebas
5. Despliegue
6. Mantenimiento

Especificación de requisitos

Los **requisitos no funcionales** conducen hacia la elección de una u otra arquitectura



Modelo de calidad de la ISO/IEC 25010

Diseño del sistema

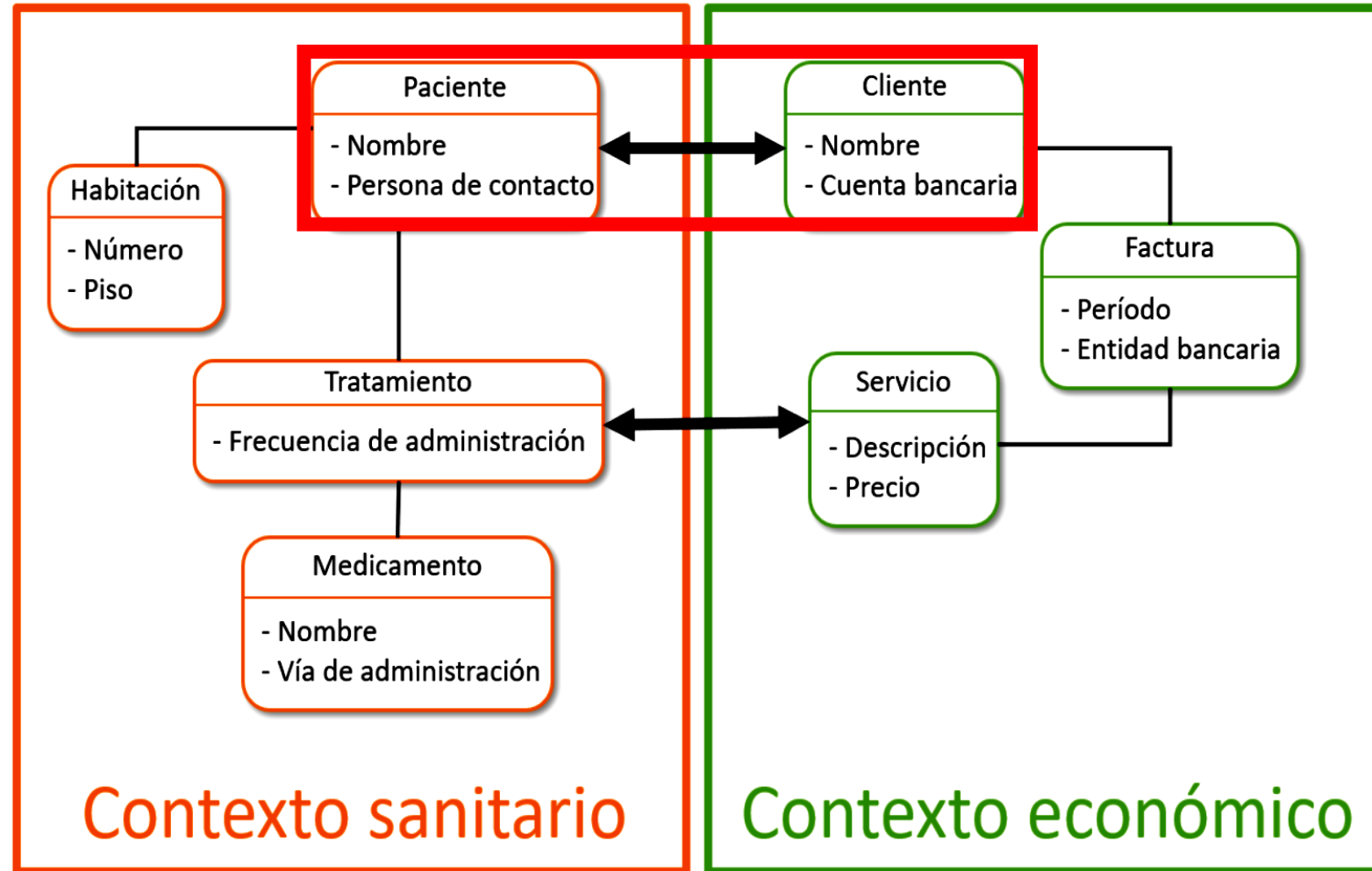
- Diseño guiado por el dominio (DDD)

Contextos delimitados



Microservicios

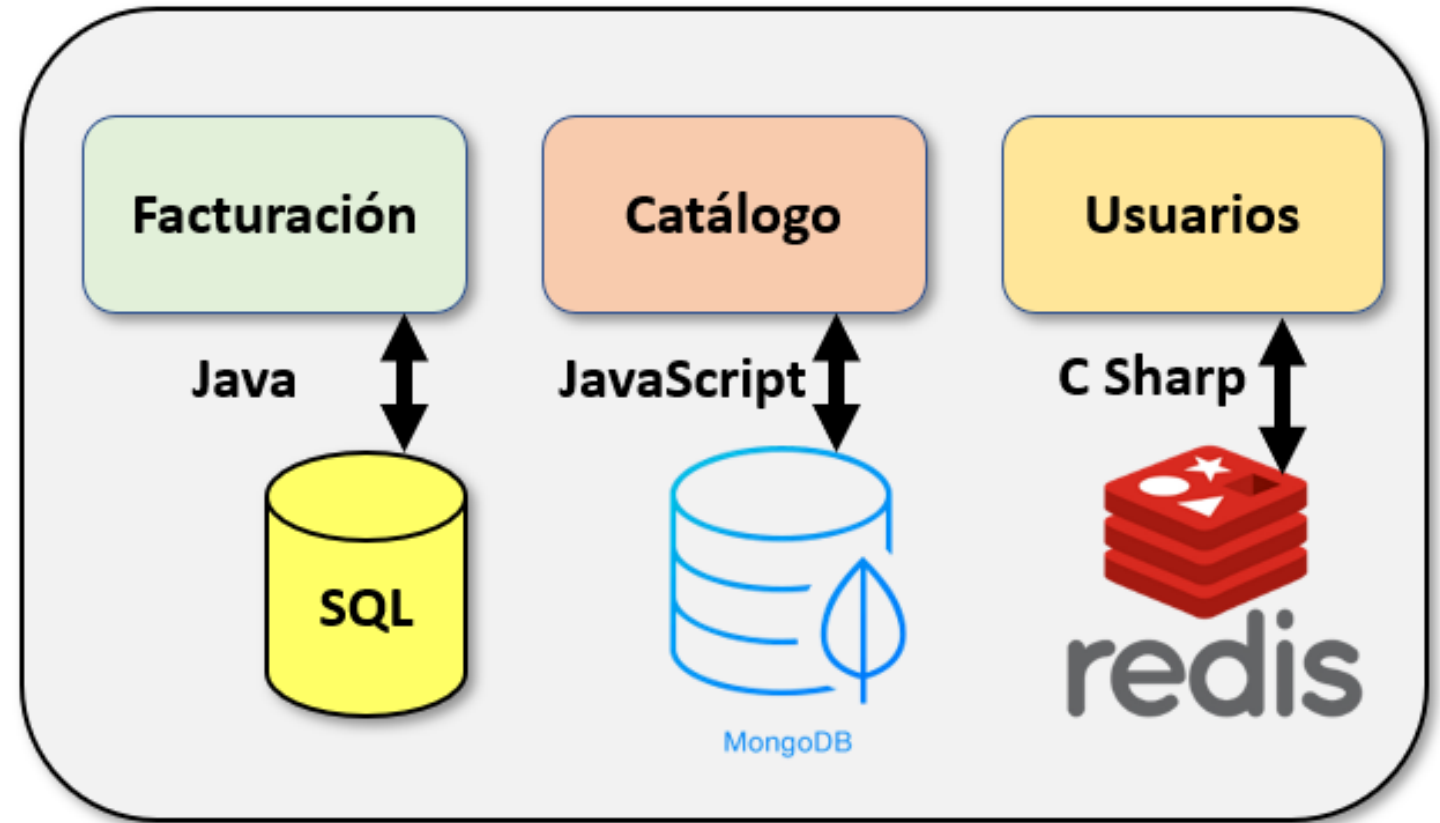
Dominio: ERP para la gestión de un hospital



Implementación del sistema

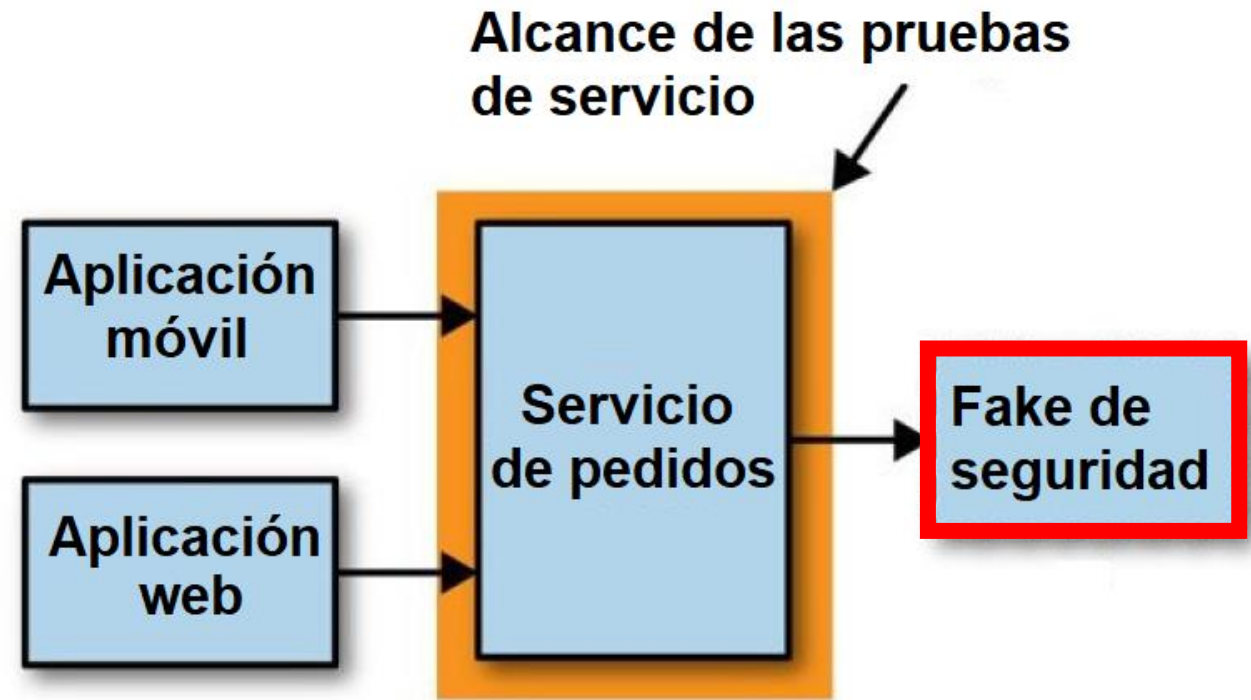
- Integración de microservicios:
 - RPC
 - REST
 - Basada en eventos

Sistema para la gestión de un hospital



Pruebas

- ↑ Facilidad de prueba
- Hacer las pruebas lo más sencillas posibles utilizando *fakes*.



Despliegue

- Máquinas virtuales:

Mayor tiempo de despliegue y consumo de recursos.

- Contenedores:

Más ligeros pero menor grado de aislamiento.

Despliegue con máquinas virtuales

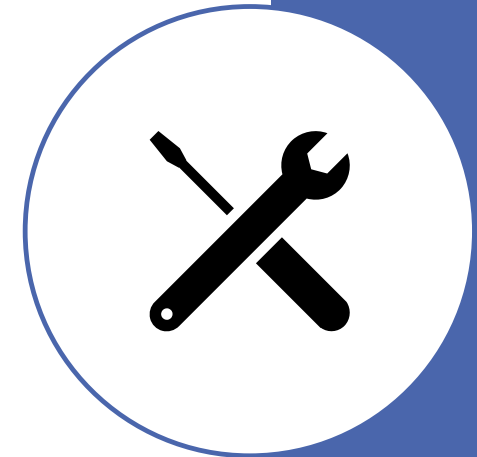


Despliegue con contenedores



Fase de mantenimiento

- ***“You build it, you run it” - Amazon***
 - El mismo equipo que implementa un microservicio realiza su mantenimiento
- Garantizar los acuerdos de nivel de servicio mediante la monitorización





Estado del arte

1. Contenedores
2. Orquestadores

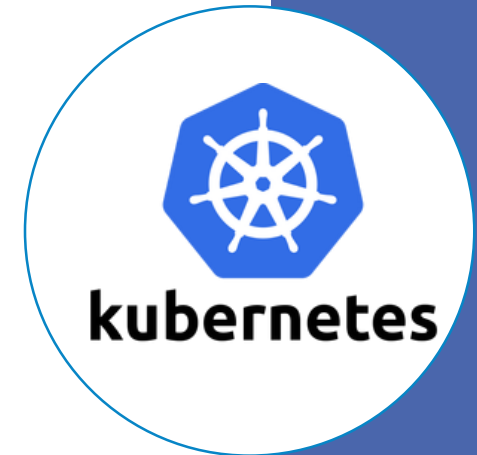
Contenedores

- **Contenedores Linux (LXC)**
 - Limitan al uso de Linux como base del entorno
- **Contenedores Docker**
 - Funcionamiento sencillo
 - Desechables y reproducibles



Orquestadores

- **Kubernetes**
 - Sencillo garantizar la disponibilidad
 - Reglas para gestionar su escalabilidad
- **Docker Swarm**
 - Orquestador nativo para los contenedores Docker

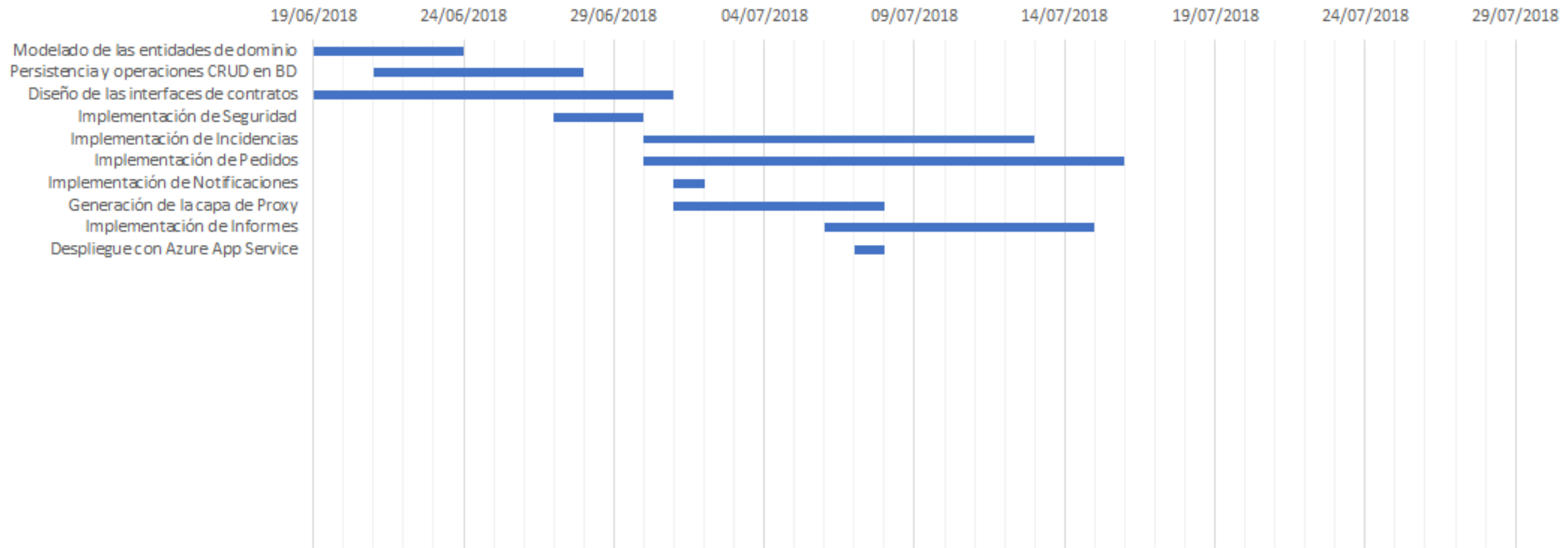




Caso de estudio

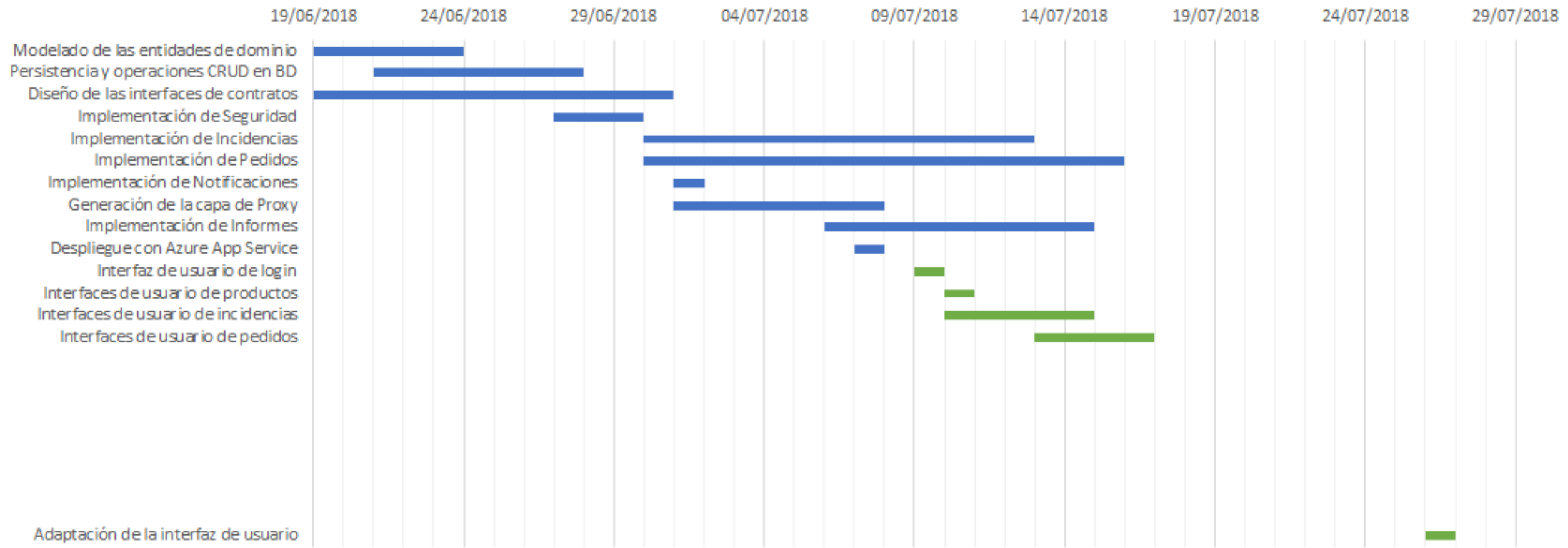
1. Plan de trabajo
2. Especificación
3. Sistema monolítico
4. Sistema basado en microservicios
5. Demostración

Cronograma del desarrollo del caso de estudio



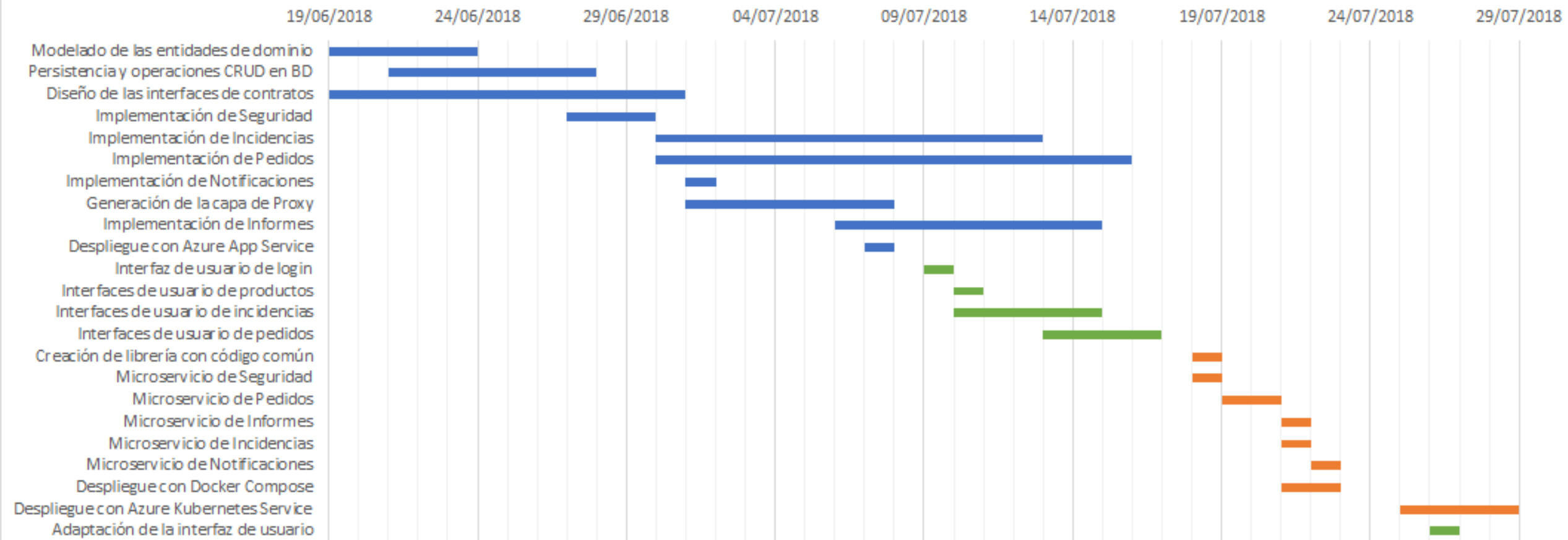
Plan de desarrollo

Cronograma del desarrollo del caso de estudio



Plan de desarrollo

Cronograma del desarrollo del caso de estudio

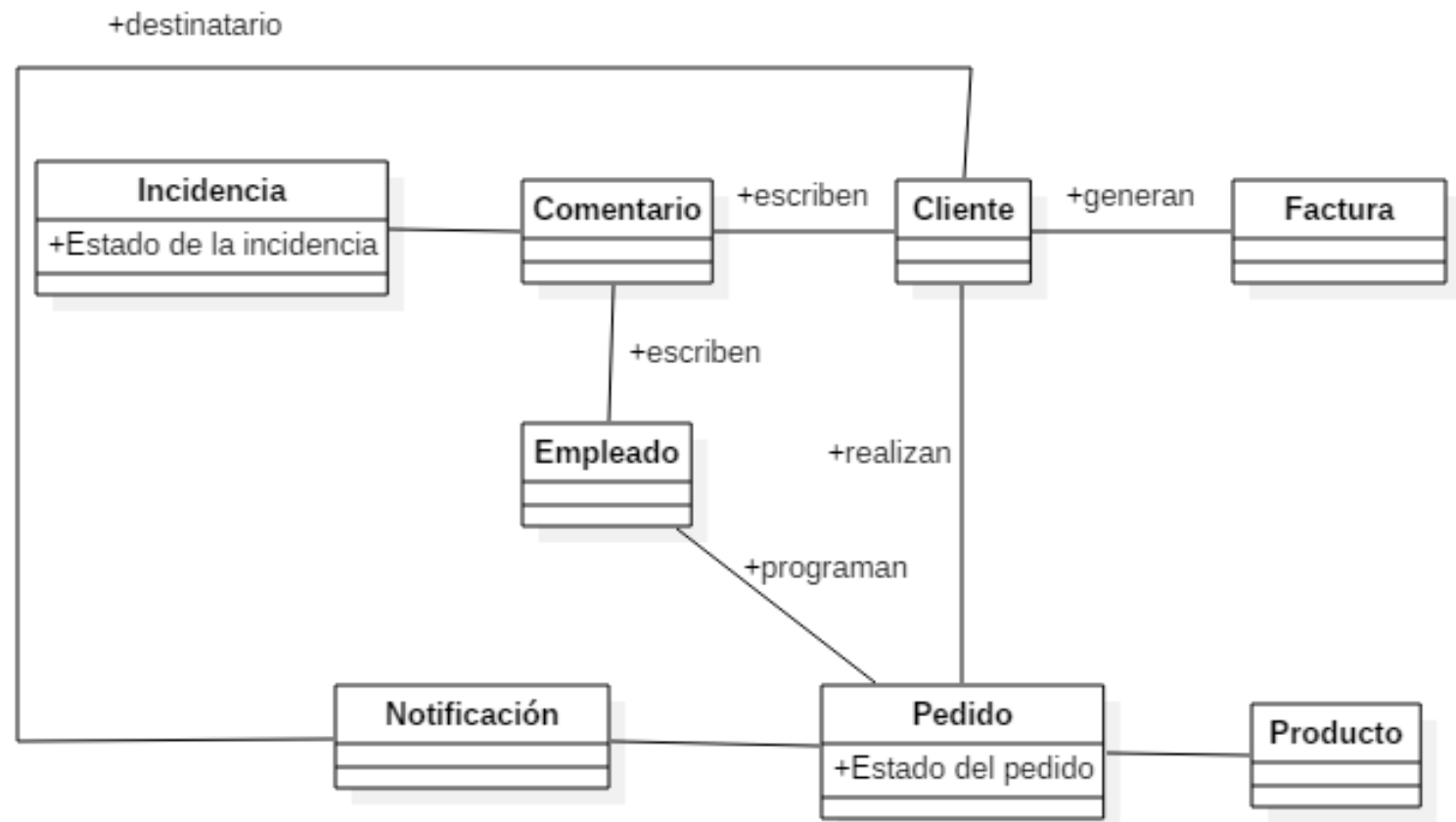


Plan de desarrollo

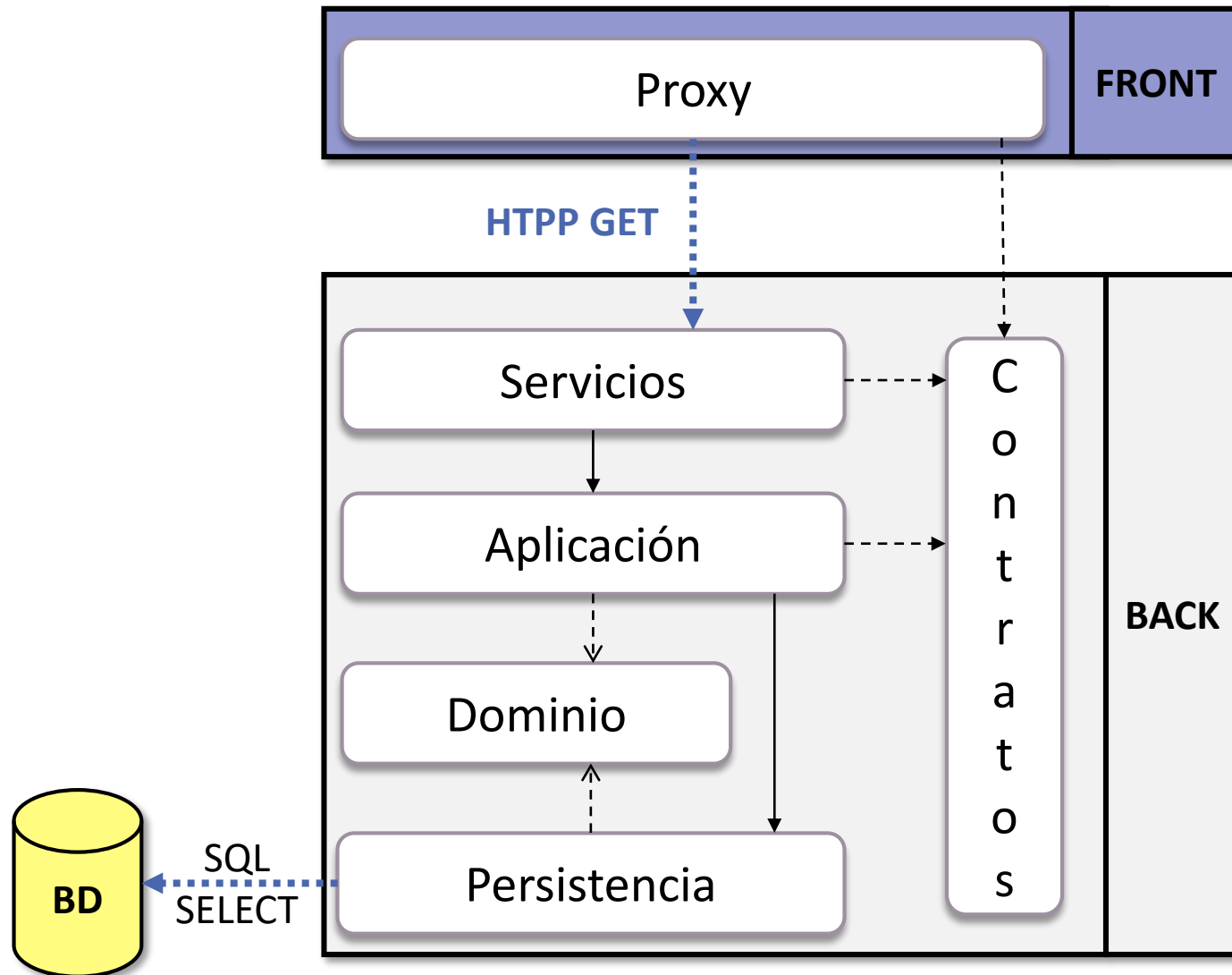
Especificación del caso de estudio

Aplicación móvil para un sistema de comercio electrónico:

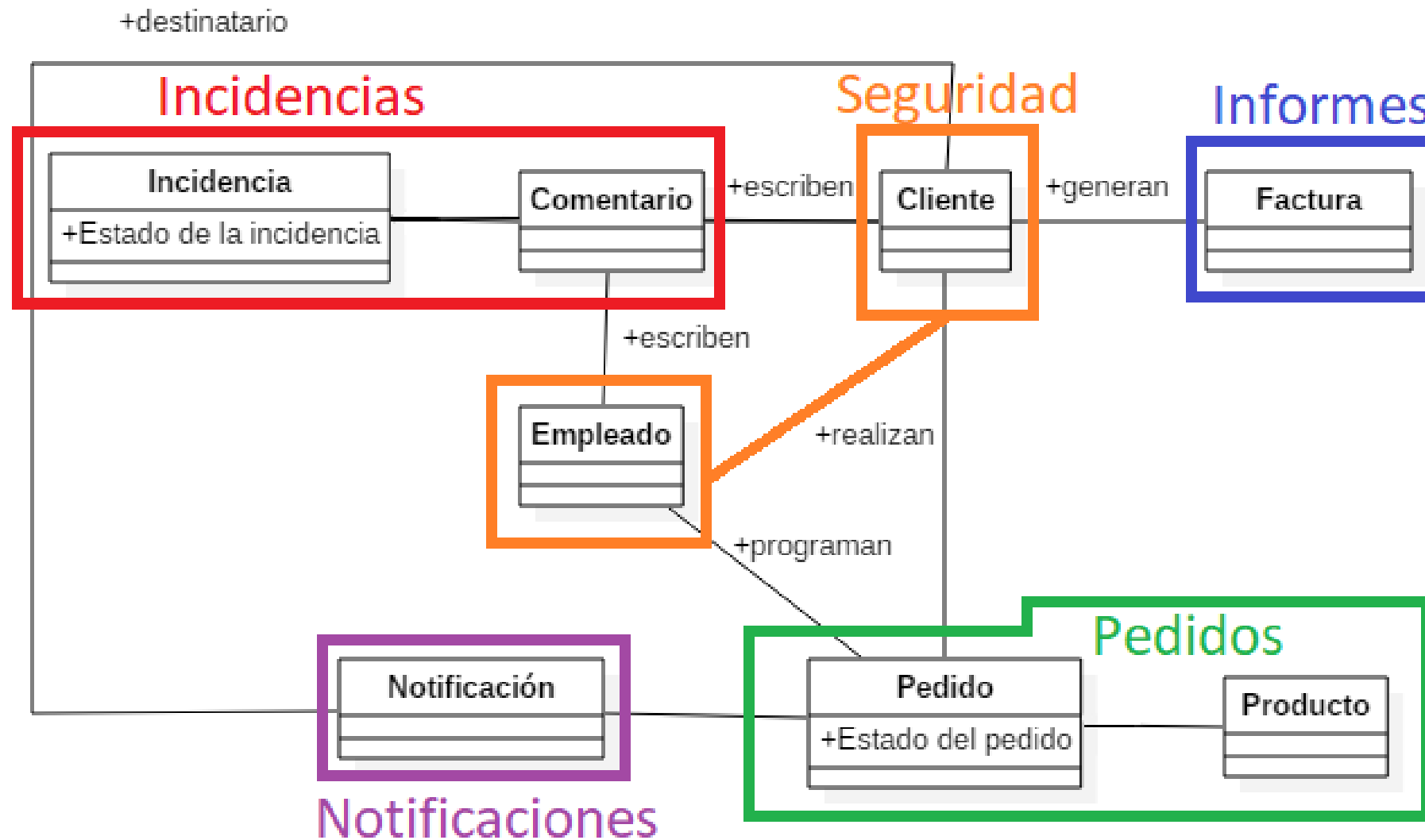
- Realizar pedidos
- Ver factura de un pedido
- Crear una incidencia

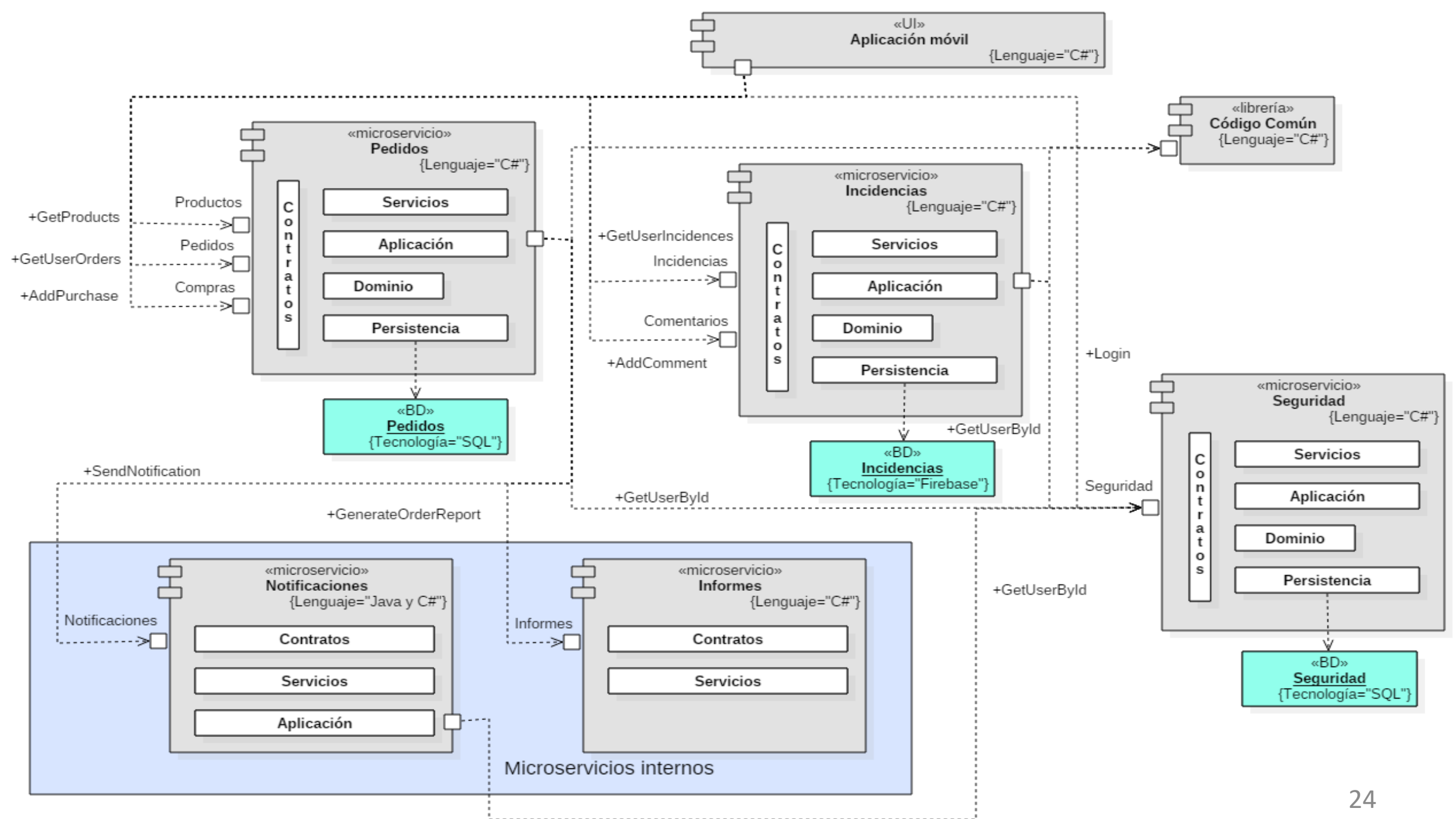


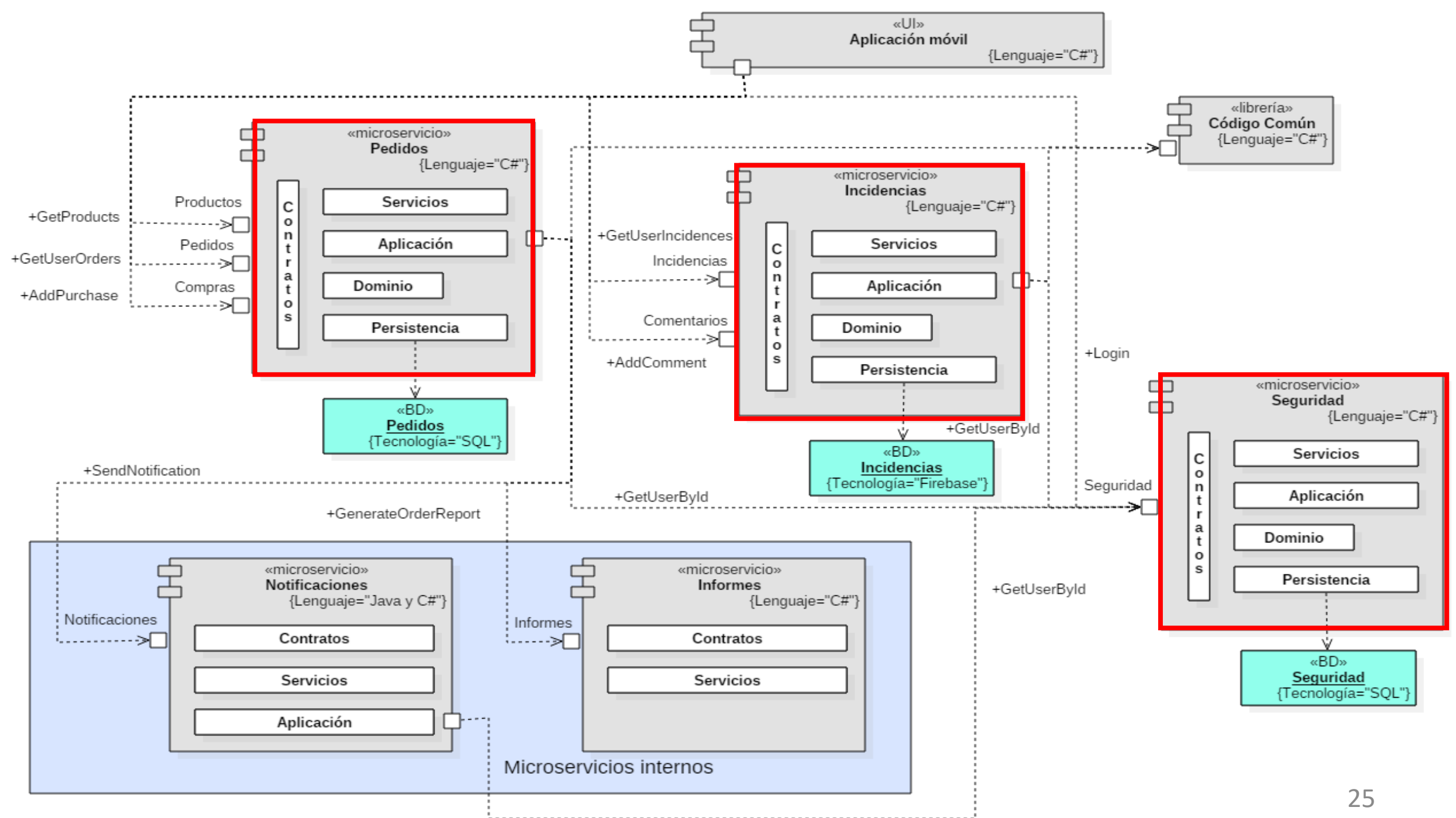
Arquitectura monolítica

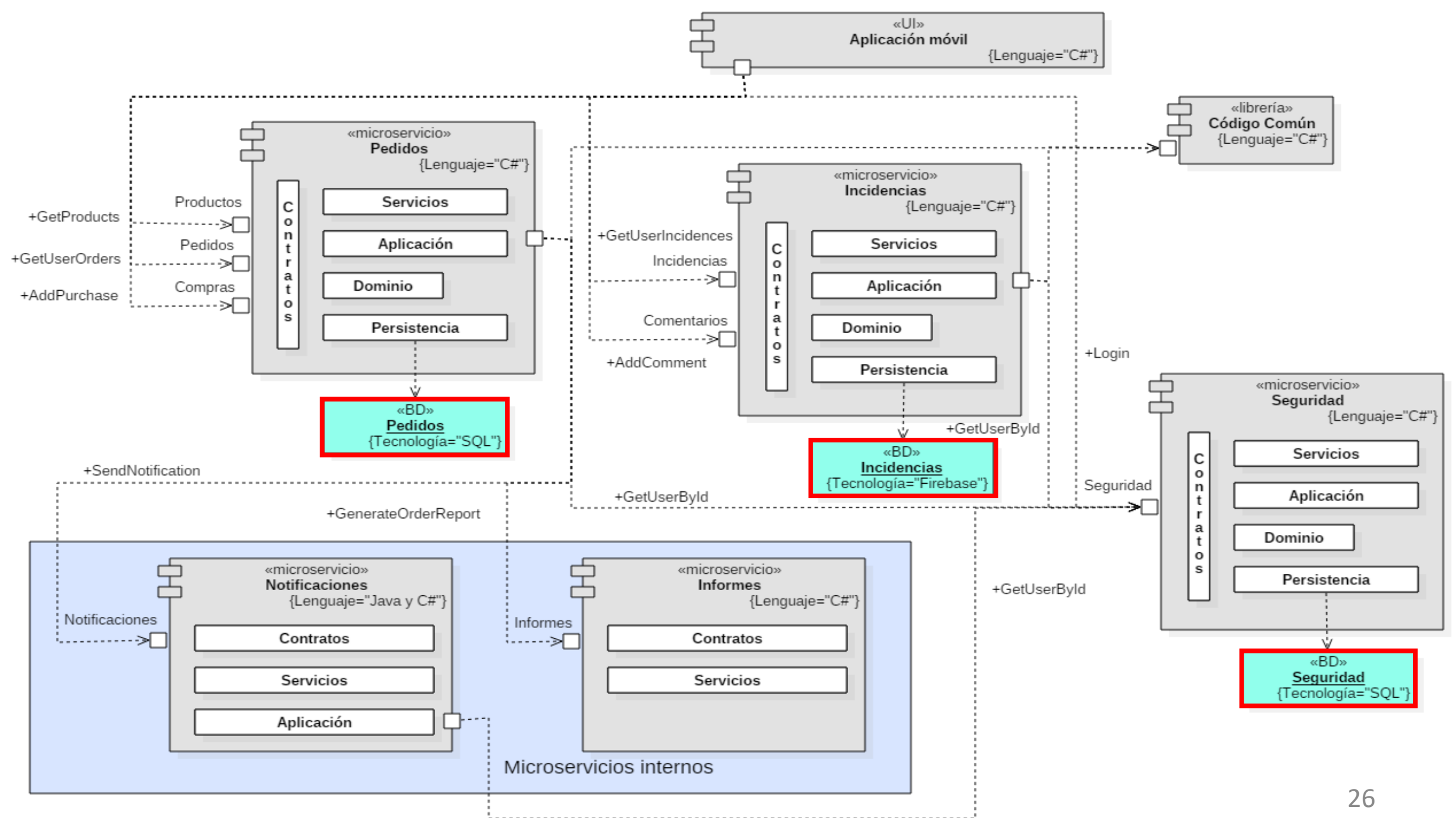


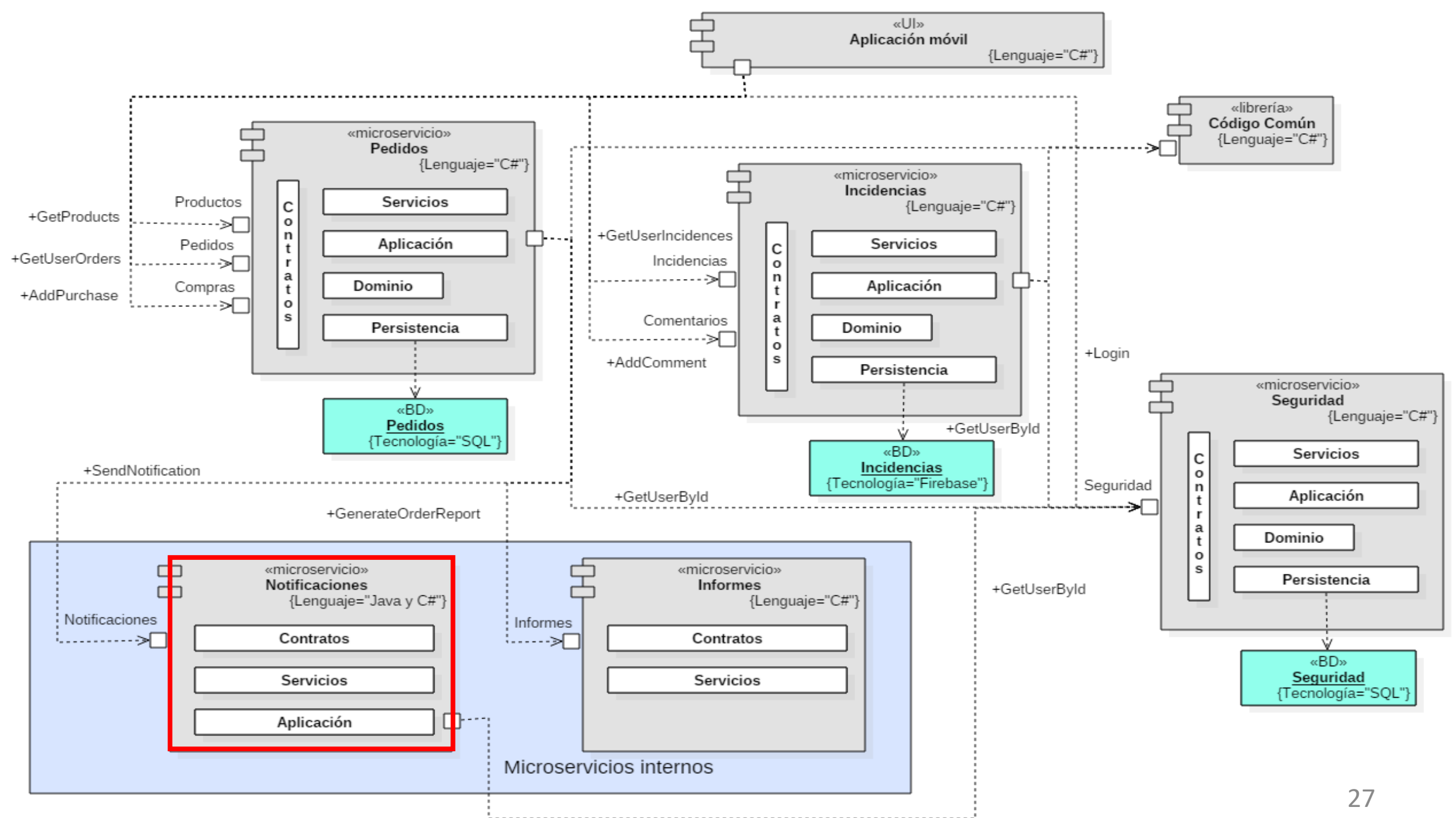
Arquitectura basada en microservicios

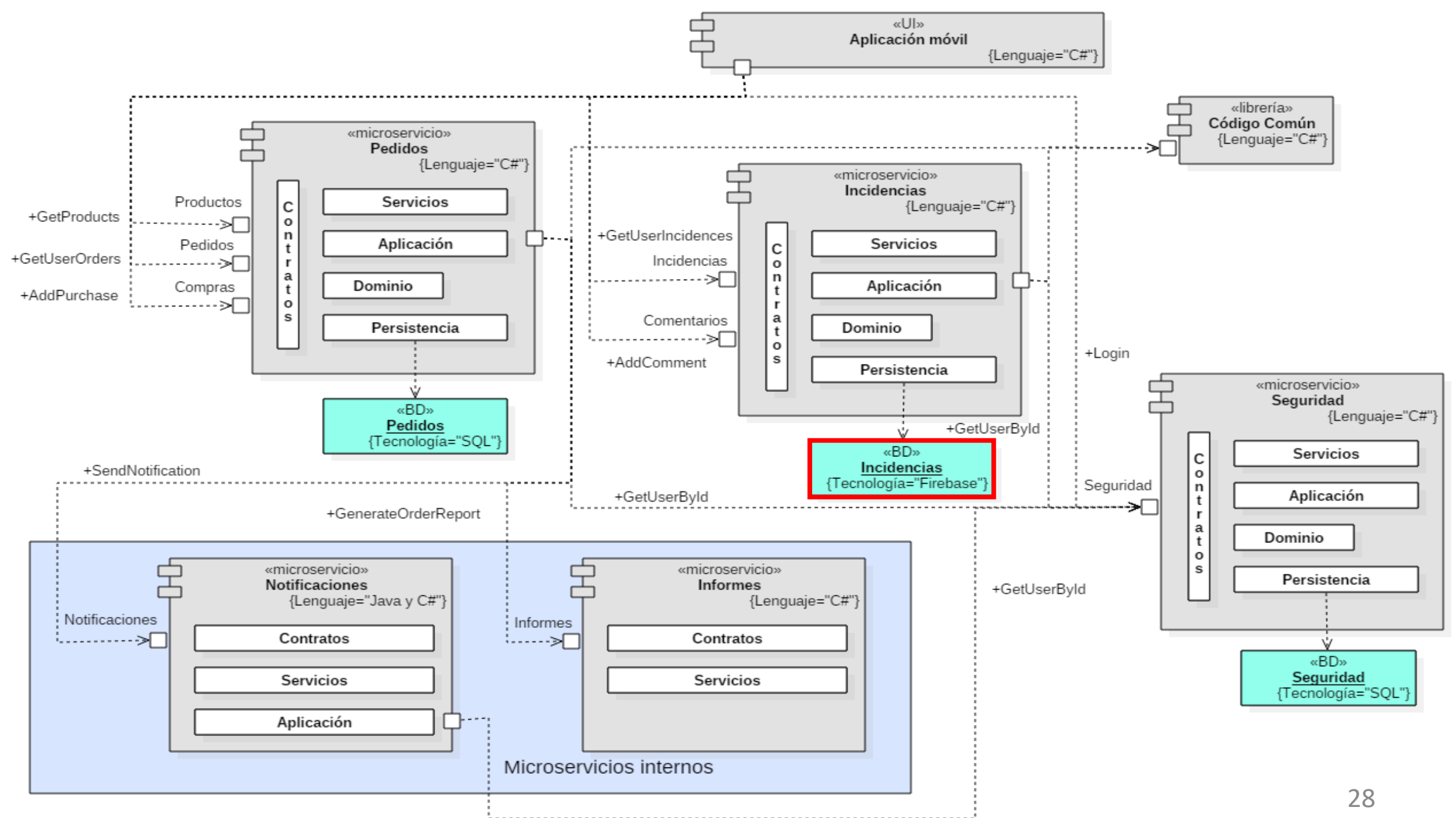


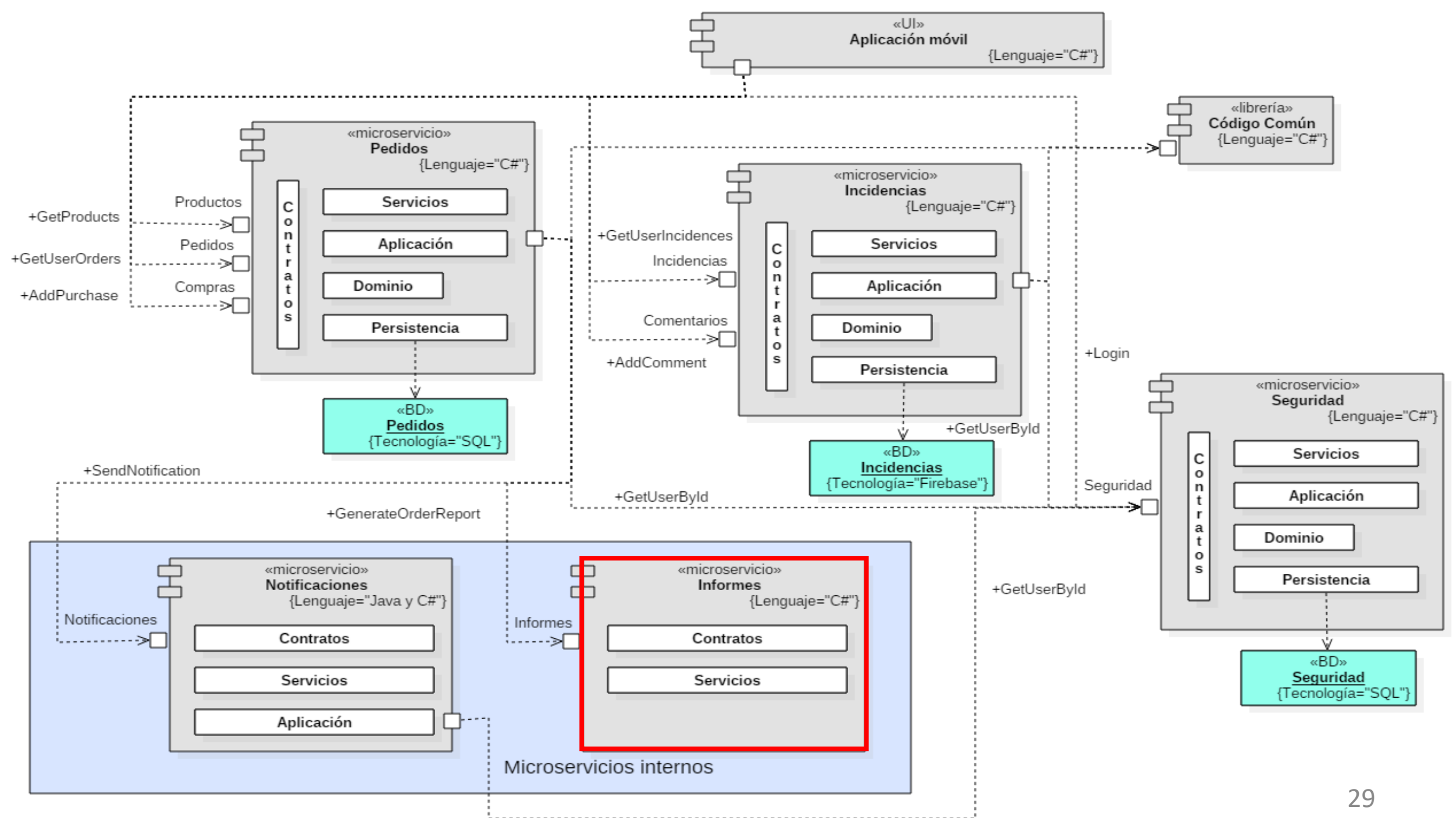














Herramientas para la construcción

- Persistencia: **Entity Framework Core**.
- Seguridad: Identity.
- Informes: Open XML PowerTools.
- Notificaciones: MailKit.
- API interactiva: **Swagger UI**.
- Generación de la capa de proxy: NSwag.
- Calidad del código: **CodeMaid** y StyleCop.
- Interfaz de usuario: **Xamarin**.
- Pruebas: NUnit.
- Despliegue: **Docker, Kubernetes y Azure**.

Demostración

Mantenimiento de las soluciones

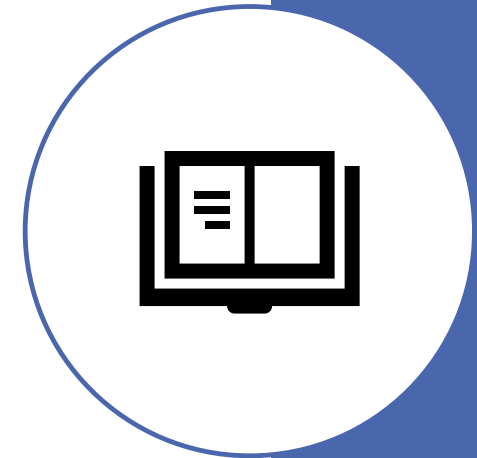
	Sistema basado en microservicios	Sistema monolítico
Mantenimiento correctivo	<p>Los defectos:</p> <ul style="list-style-type: none">• Se localizan en un único microservicio.• Son difíciles de depurar si involucra a más de un servicio.	<ul style="list-style-type: none">• Difíciles de localizar.• Más fácil de depurar la solución para encontrar el defecto.
Mantenimiento perfectivo	<p>Los nuevos requisitos:</p> <ul style="list-style-type: none">• Encajan dentro de un microservicio.• Dan lugar a nuevos microservicios.• Replantan la descomposición del sistema.	<ul style="list-style-type: none">• Los nuevos requisitos añaden complejidad al sistema.• Hacen que el futuro mantenimiento sea más complejo.
Mantenimiento adaptativo	<p>Los cambios para adaptar el sistema:</p> <ul style="list-style-type: none">• Afectan a solo una porción del sistema.• Pueden abordarse de forma incremental.	<ul style="list-style-type: none">• Los cambios afectan al sistema en su totalidad.• No pueden abordarse de forma incremental.

Evaluación de requisitos no funcionales

	Sistema basado en microservicios	Sistema monolítico
Disponibilidad	Se garantiza frente a algunas situaciones gracias al uso de Kubernetes.	No se ha implementado ningún mecanismo.
Tolerancia a fallos	<ul style="list-style-type: none">• Se asume que cualquier servicio puede fallar.• Un servicio caído no deja inoperativo al resto del sistema.• Cada microservicio tiene su propia base de datos. Así, no hay un único punto de fallo.	<ul style="list-style-type: none">• El fallo de un módulo puede suponer que todo el sistema se encuentre inoperativo.• Existe una única base de datos, por lo que existe un único punto de fallo en los datos.
Utilización de recursos	No se han apreciado grandes diferencias. Teóricamente, estas se perciben conforme el sistema escala.	
Capacidad de ser reemplazado	<ul style="list-style-type: none">• Tiempo para reemplazar un microservicio: 2 semanas.• Se puede abordar de forma incremental.	<ul style="list-style-type: none">• Tiempo para reemplazar el sistema: 1 mes.• NO se puede abordar de forma incremental.

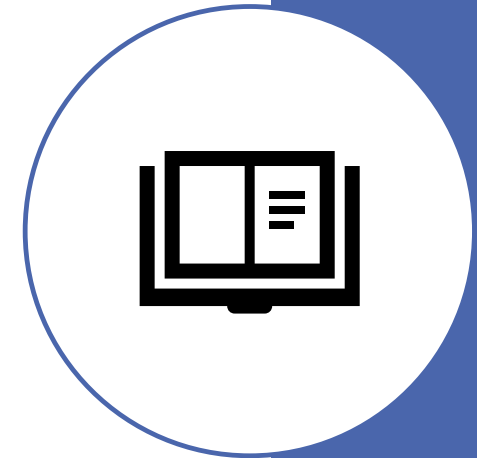
Conclusiones y trabajos futuros

- ✓ Desarrollo **satisfactorio** siguiendo ambas arquitecturas
- ✓ Desarrollo más desafiante en las actividades de diseño y despliegue en una solución basada en **microservicios**
- ✓ Mantenimiento más **simple** en un sistema basado en microservicios
- ✓ **Ventaja** de los sistemas basados en microservicios para satisfacer los RNFs bajo estudio



Conclusiones y trabajos futuros ...

- ✓ Aplicación de los conocimientos obtenidos en asignaturas como Proceso de Software (**PSW**) o Tecnología de Sistemas de Información en la Red (**TSR**)
- ✓ Mayor desempeño profesional
- ✓ **Líneas de trabajo futuro:** aplicación de un modelo de calidad



Desarrollo de software basado en microservicios: un caso de estudio para evaluar sus ventajas e inconvenientes



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Escola Tècnica
Superior d'Enginyeria
Informàtica



etsinf

Autor: Víctor Alberto Iranzo Jiménez

Tutores: Patricio Orlando Letelier Torres

María Carmen Penadés Gramage

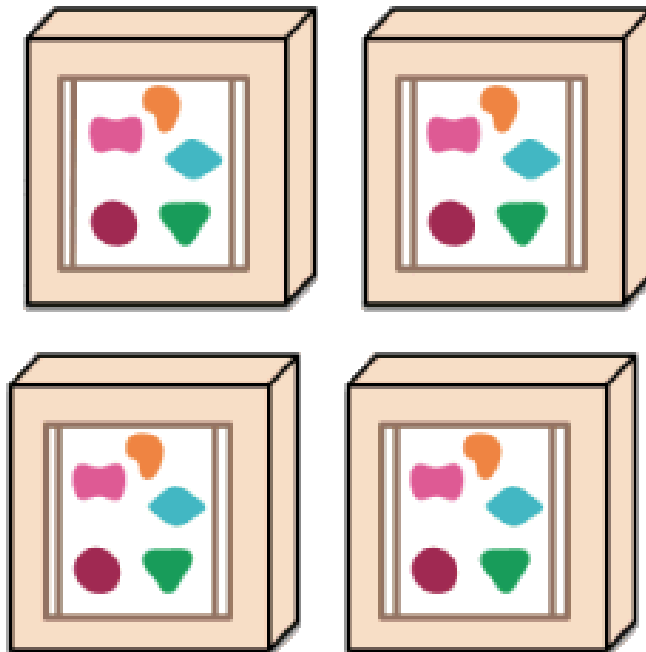
Jueves, 20 de Septiembre de 2018

Comparación de los sistemas cuando escalan

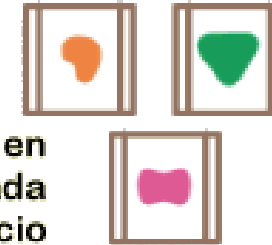
Una aplicación monolítica pone todos sus componentes en el mismo proceso



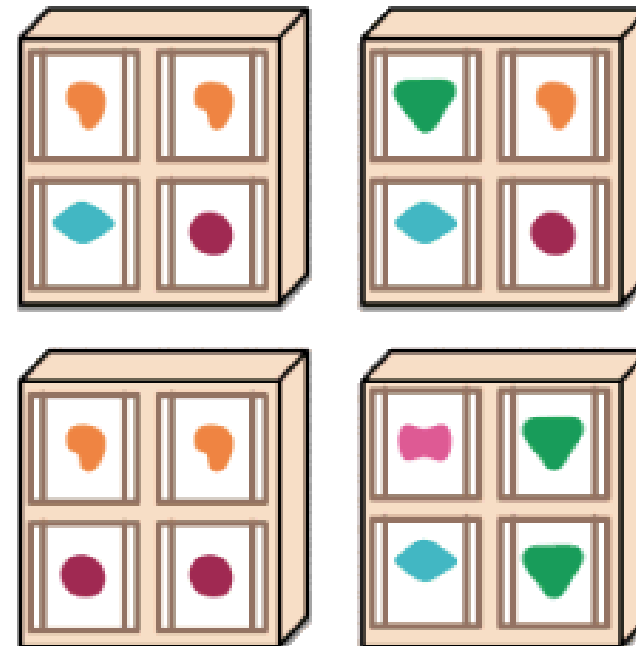
Y escala replicando el proceso a través de múltiples servidores



Una arquitectura basada en microservicios pone cada funcionalidad en un servicio

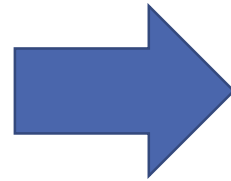
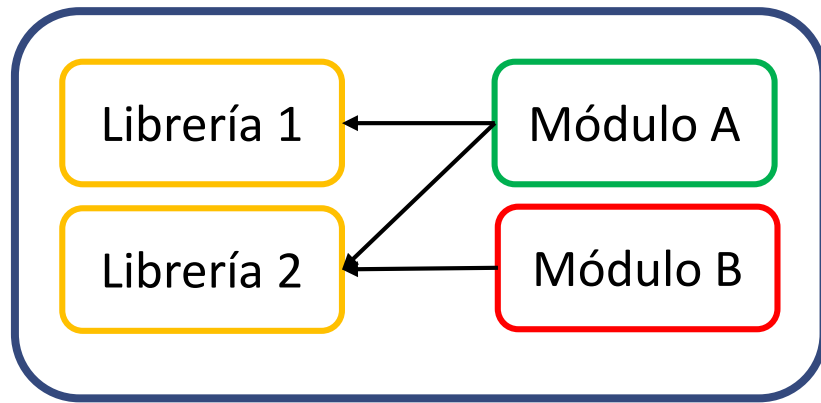


Y escala distribuyendo y replicando los servicios según sea necesario

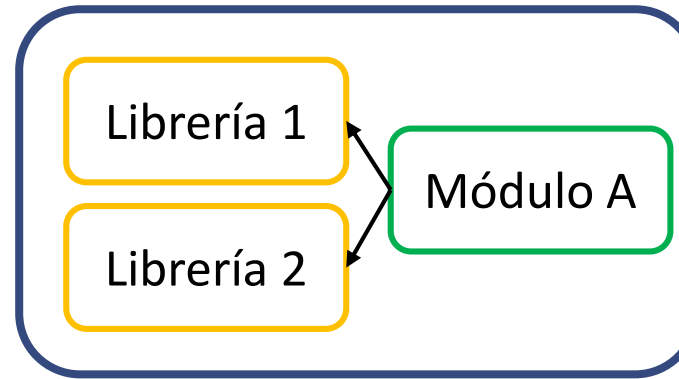


Comparación de los sistemas cuando escalan

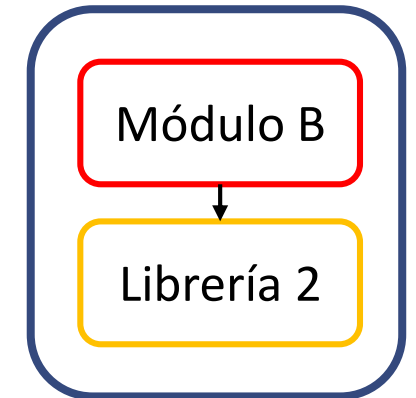
Sistema monolítico



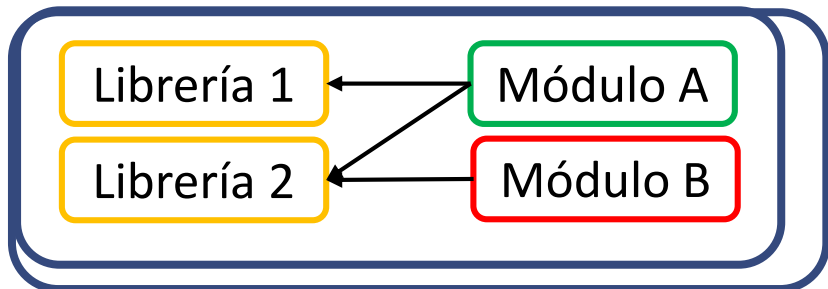
Microservicio A



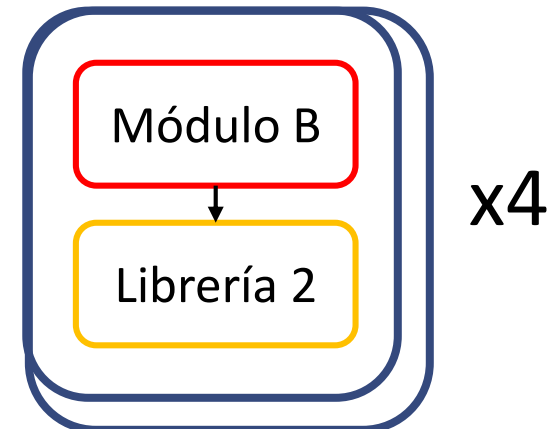
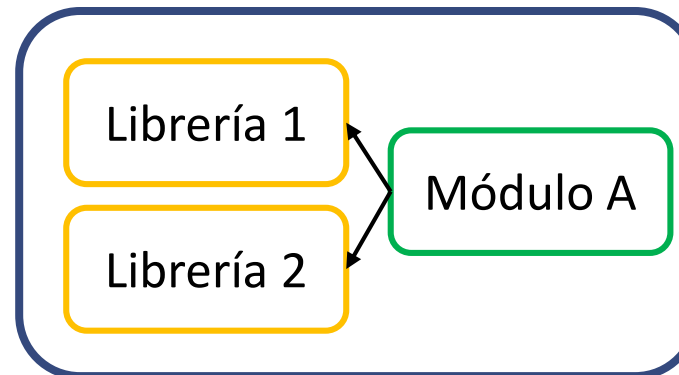
Microservicio B



Necesitas escalar 4 veces el módulo B.



x4 vs



Demostración

- https://youtu.be/-m_pTzhl3V8

Evolución de los microservicios

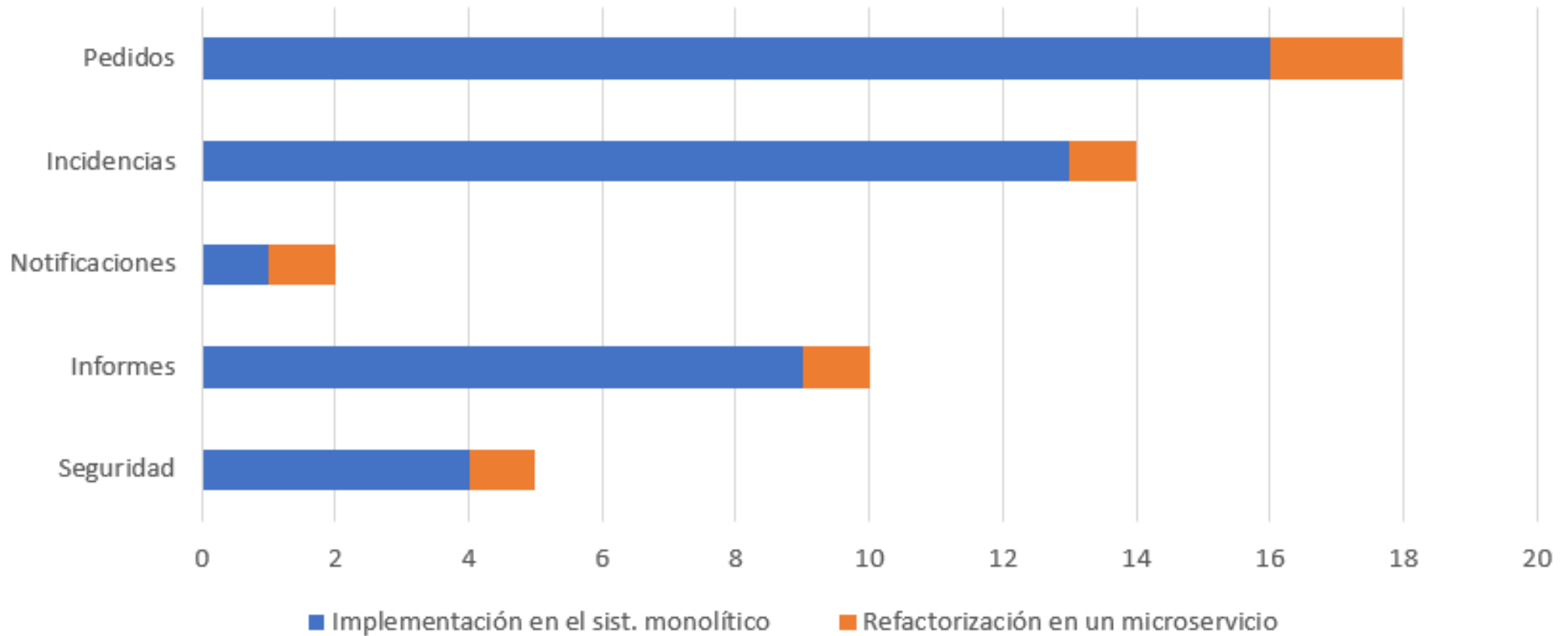
Los microservicios evolucionan y se versionan de forma independiente

```
<Project>
  <PropertyGroup>
    <Common>1.0.5</Common>
    <Security>1.0.2</Security>
    <Orders>1.0.0</Orders>
    <Reports>1.0.0</Reports>
    <Incidences>1.0.0</Incidences>
    <Notifications>1.0.1</Notifications>
  </PropertyGroup>
</Project>
```

Capacidad para ser reemplazado

Hipótesis: el tiempo para desarrollar un componente es el mismo que el de reemplazarlo.





Capacidad para ser
reemplazado ...

- Media de tiempo para desarrollar un microservicio: **10 días**.
- No se incluyen la realización de los formularios asociados a cada microservicio.

Capacidad para ser reemplazado ...

X Desarrollo de microservicios es más costoso: no se puede afirmar que la suma de implementar un componente en el sistema monolítico y refactorizarlo sea igual al tiempo necesario para implementarlo directamente como un microservicio.

- Integración de los microservicios
- Diseñar invocaciones interprocedurales

X Al reemplazar un microservicio:

- Interfaces ya establecidas.
- Implementación ya refinada.

Conclusión: hace falta realizar una evaluación más formal.

Adaptación de la UI para usar microservicios

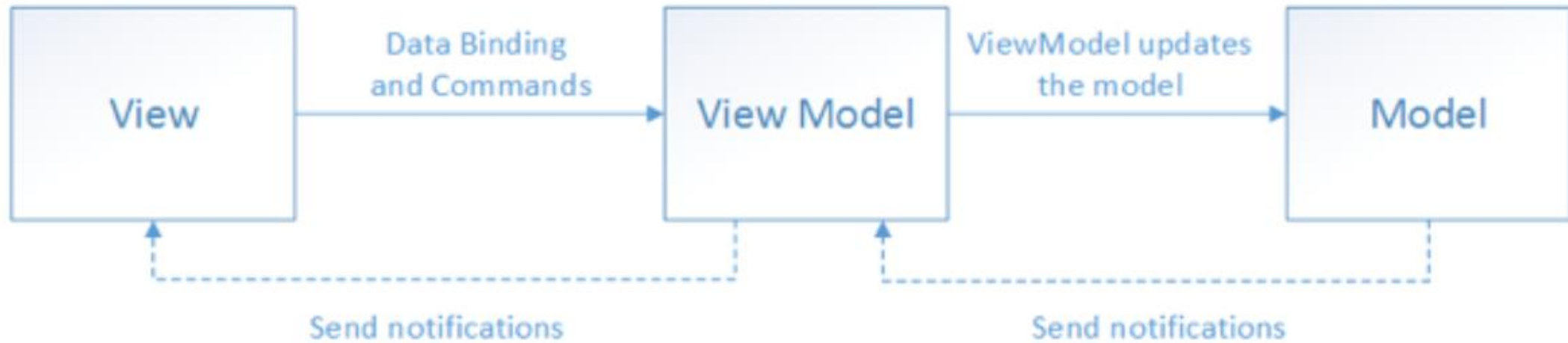
```
using global::Xamarin.Forms;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
- using Shop.Backend.Monolithic.Proxy;
+ using Shop.Backend.Microservices.Incidents.Proxy;
+ using Shop.Backend.Microservices.Orders.Proxy;
+ using Shop.Backend.Microservices.Security.Proxy;
using Shop.Frontend.Xamarin.Components.Resources;
using Shop.Frontend.Xamarin.Helpers;
using Shop.Frontend.Xamarin.Interfaces;

@@ -27,8 +29,14 @@ public App()
```

```
    this.serviceCollection = new ServiceCollection();
```

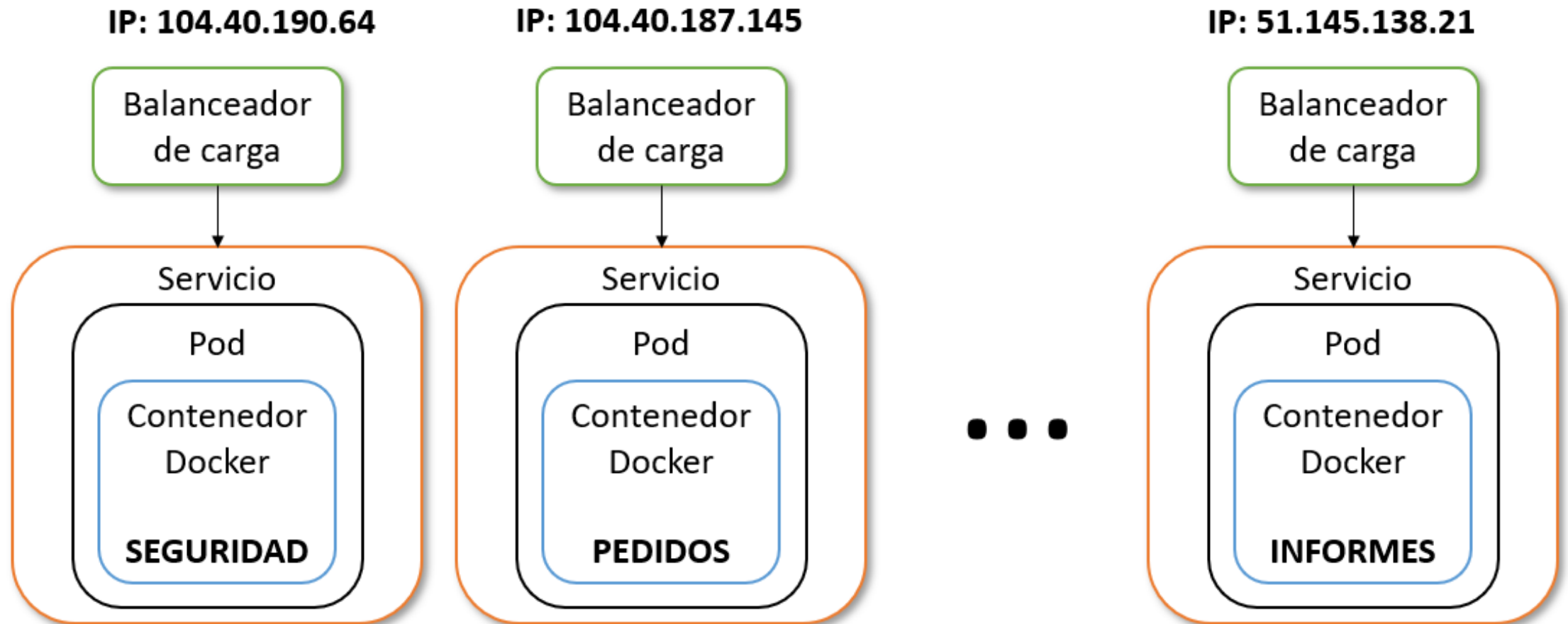
```
-    this.serviceCollection.AddProxyDependencies(this.configuration);
-    this.serviceCollection.RegisterClients(new HttpClient(), this.configuration);
+    this.serviceCollection.AddSecurityProxy(this.configuration);
+    this.serviceCollection.RegisterSecurityClient(new HttpClient(), this.configuration);
+
+    this.serviceCollection.AddOrdersProxy(this.configuration);
+    this.serviceCollection.RegisterOrdersClients(new HttpClient(), this.configuration);
+
+    this.serviceCollection.AddIncidentsProxy(this.configuration);
+    this.serviceCollection.RegisterIncidentsClients(new HttpClient(), this.configuration);
```

MVVM

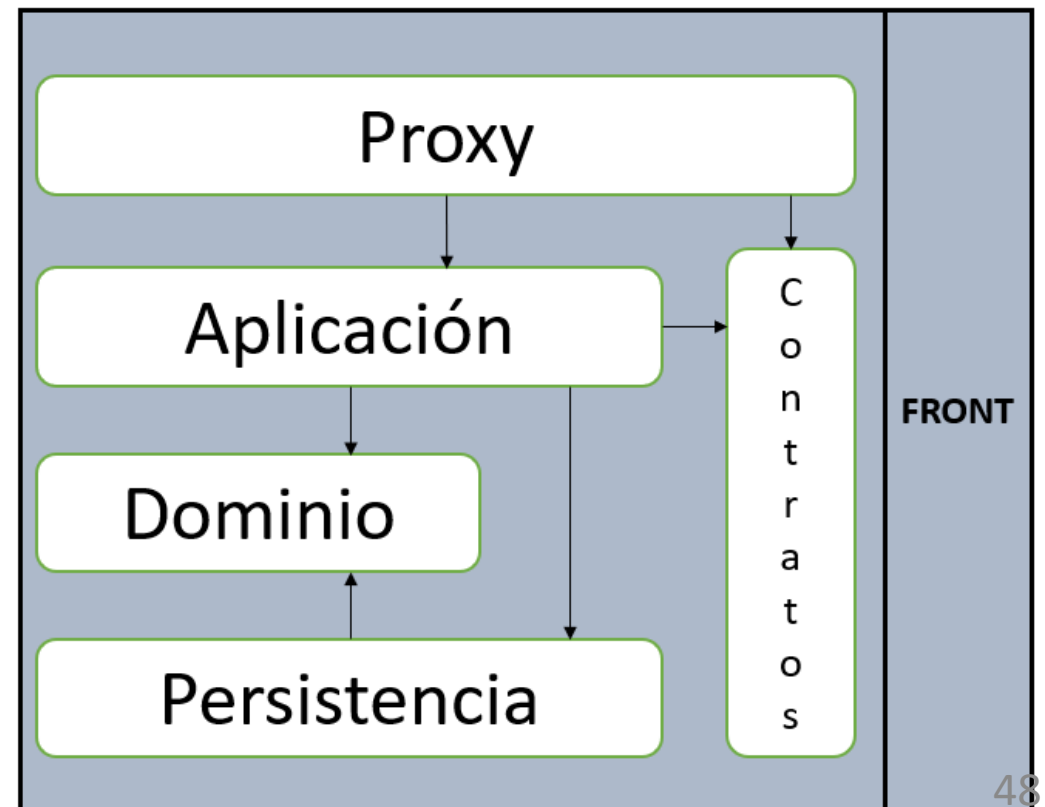
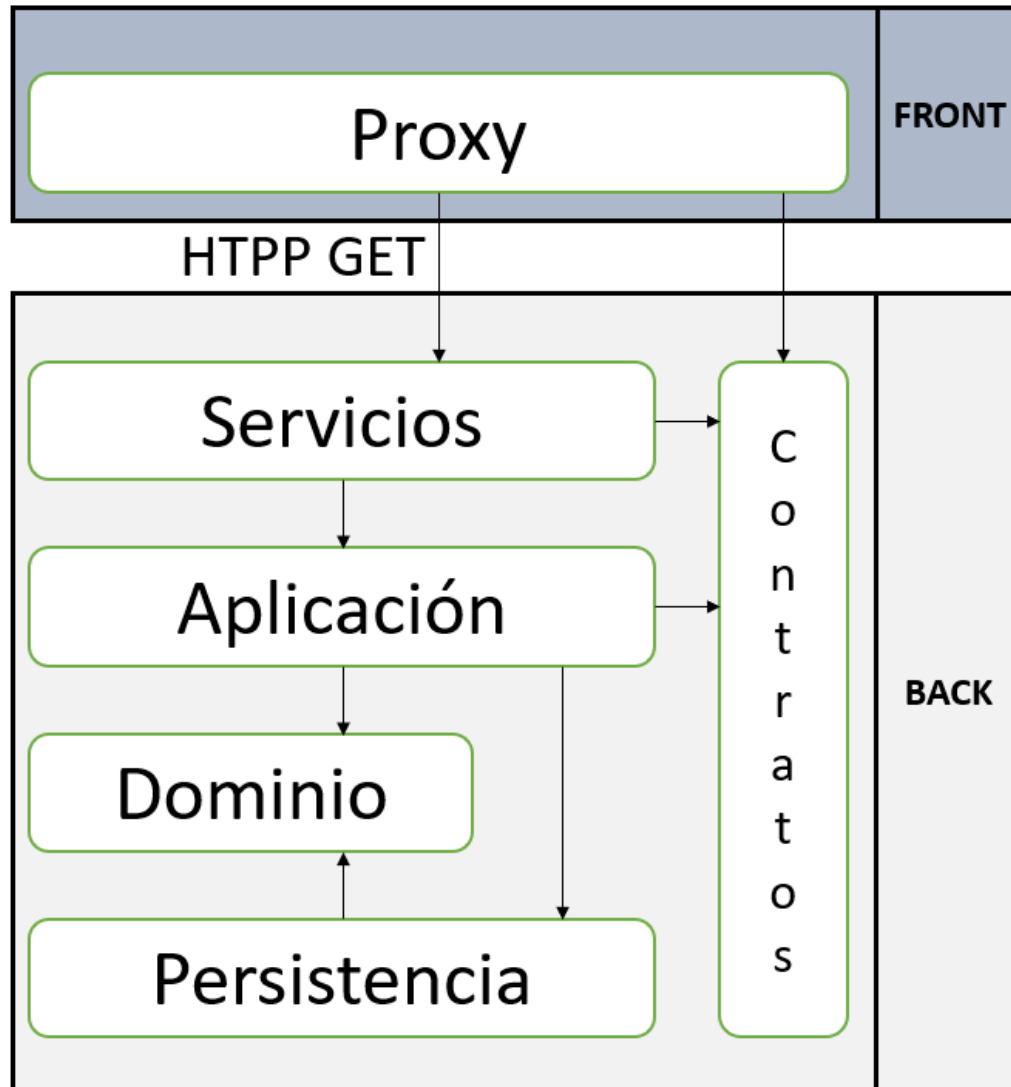


- El **modelo de la vista** aísla el modelo de la vista.
- **Vista**: define la apariencia de lo que el usuario ve. En Xamarin, una Page.
- **Modelo la vista**: define propiedades y comandos con los que la vista puede hacer bindings. Coordina la interacción con cualquier clase del dominio. Los datos están preparados para ser consumidos por la vista.
- **Modelo**: representaciones de las entidades del dominio, como un DTO.

Despliegue en producción



Uso del proxy



Pirámide de Cohn

Mayor alcance
Mayor confianza



Mayor velocidad
Mayor aislamiento

Orderbc16688c... 4G 14% 22:03

Order report

Client: Victor

Status of the order: Confirmed

Purchases:

Number of articles: 3

Product Name:	3D Printer - BQ Hephestos 2		
Product description:	The 3d printer that can be assembled in less than an hour: A DIY printer with professional results: Hephestos 2 is our second DIY (Do it yourself) printer. Hephestos is based on the Prusa i3, one of the most popular models in the RepRap community. With the latest version, we have evolved it to create a professional printer, while maintaining the spirit of the original. We have opted for a new design that enables us to achieve a larger printing volume, reaching (X) 210mm x (Y) 297mm x (Z) 220mm.		
Quantity:	1	Unitary Price:	749 €
		Total Price:	749 €

Product Name:	Gear S3 Classic		
Product description:	The Gear S3 looks and feels natural on your wrist and when you use it. Like a traditional watch. Only much more capable. A forward-thinking choice that offers a touch of luxury. The Gear S3 sets you free in the most natural way. You only have to turn the bezel to respond to calls, to read messages, to go to that app you need. Answer a call, turn up the volume or turn off the alarm. Turn the bezel to scroll through apps, messages and long text. It could not be easier to do what you want, to get to what you want. The Gear S3 has a built-in speaker that lets you make or take calls right away without ever having to remove your phone from your pocket.		
Quantity:	2	Unitary Price:	299 €
		Total Price:	598 €




Product Name:	Iphone X 64GB		
Product description:	Our vision has always been to create an iPhone that is entirely screen. One so immersive the device itself disappears into the experience. And so intelligent it can respond to a tap, your voice, and even a glance. With iPhone X, that vision is now a reality. Say hello to the future. Design and Display: It's all-screen. Super retina display: With iPhone X, the device is the display. An all-new 5.8-inch Super Retina screen fills the hand and dazzles the eyes.		
Quantity:	1	Unitary Price:	1159 €
		Total Price:	1159 €

Summary:

Total without VAT:	2506 €
VAT:	21.5 %
Total with VAT:	3044.79 €

My Shop

Order report

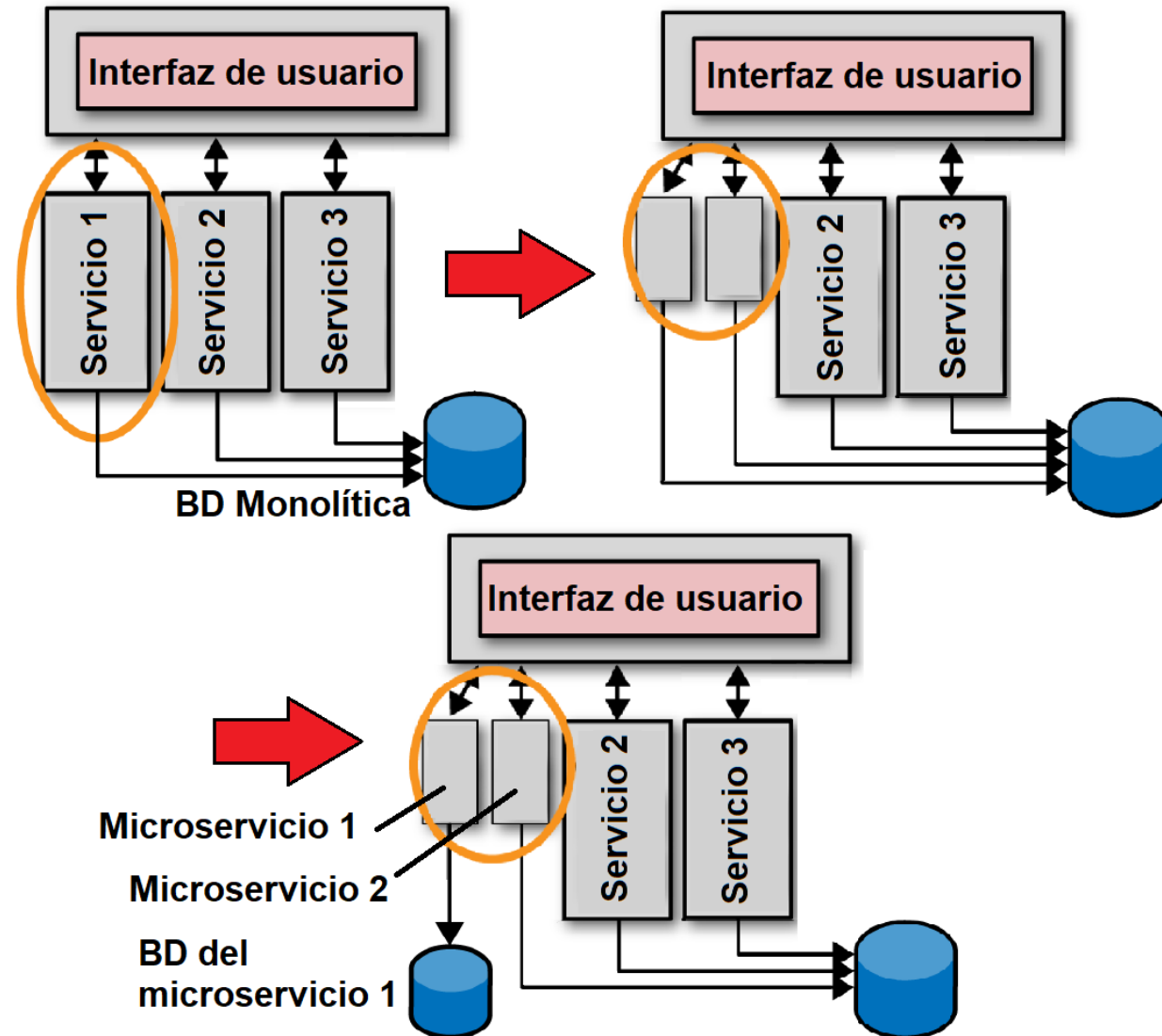
<p>3D Printer - BQ Hephestos 2</p> <p>Unitary price: 749 €</p>	<p>1</p>	
<p>Gear S3 Classic</p> <p>Unitary price: 299 €</p>	<p>2</p>	
<p>Iphone X 64GB</p> <p>Unitary price: 1,159 €</p>	<p>1</p>	

ADD MORE PRODUCTS

CONFIRM ORDER

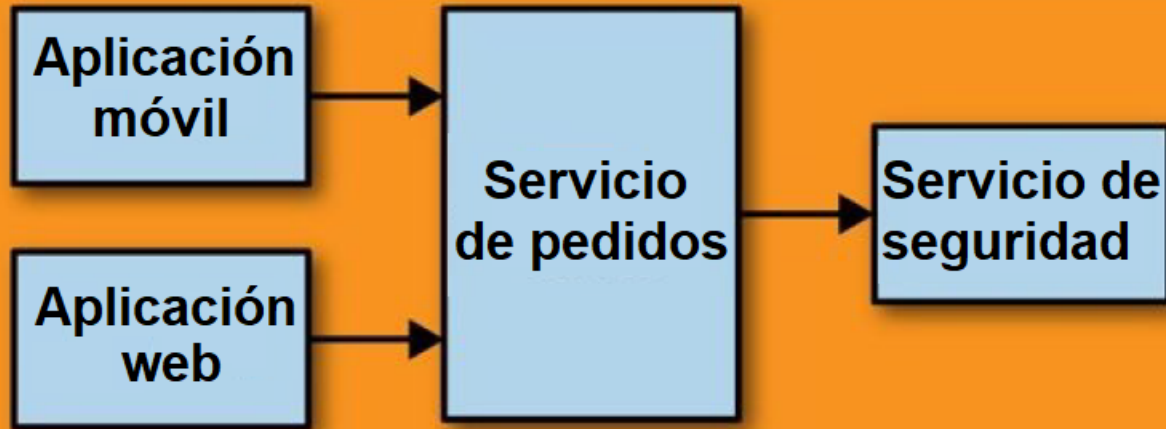
50

Migración a un sistema basado en microservicios

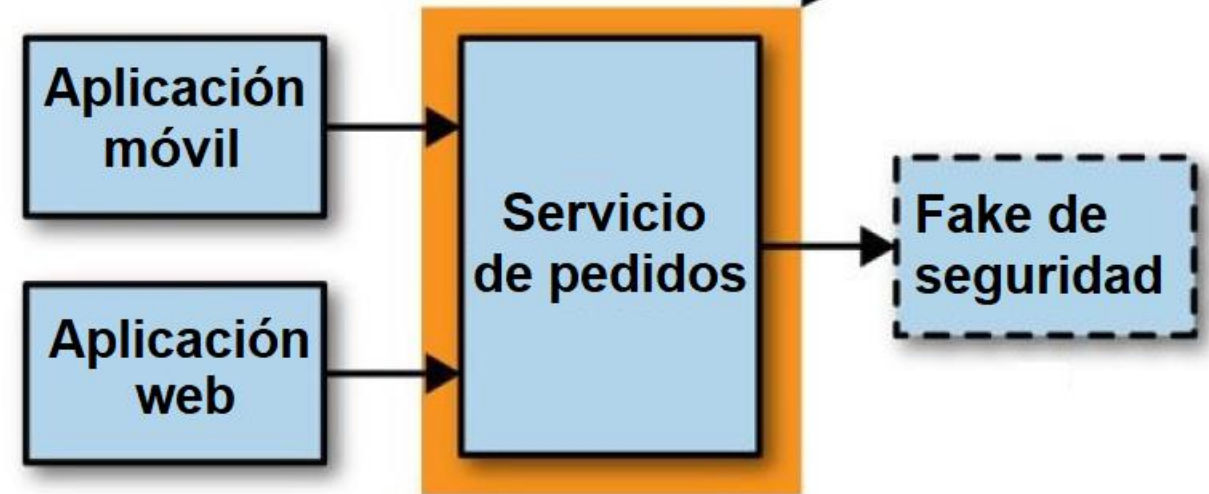


Pruebas en los microservicios

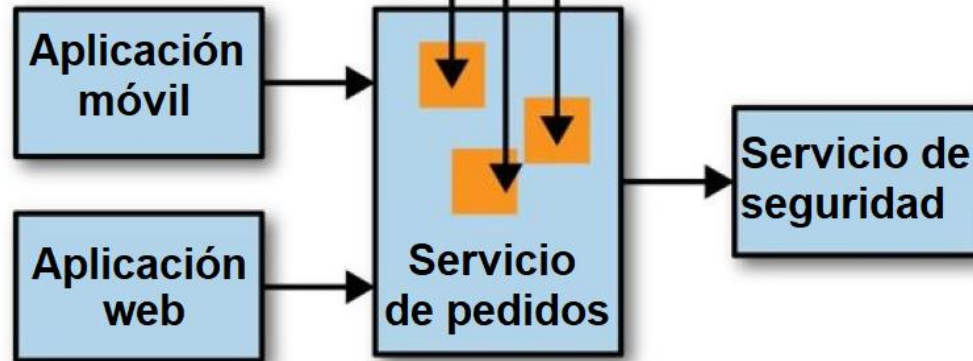
Alcance de las pruebas de extremo a extremo



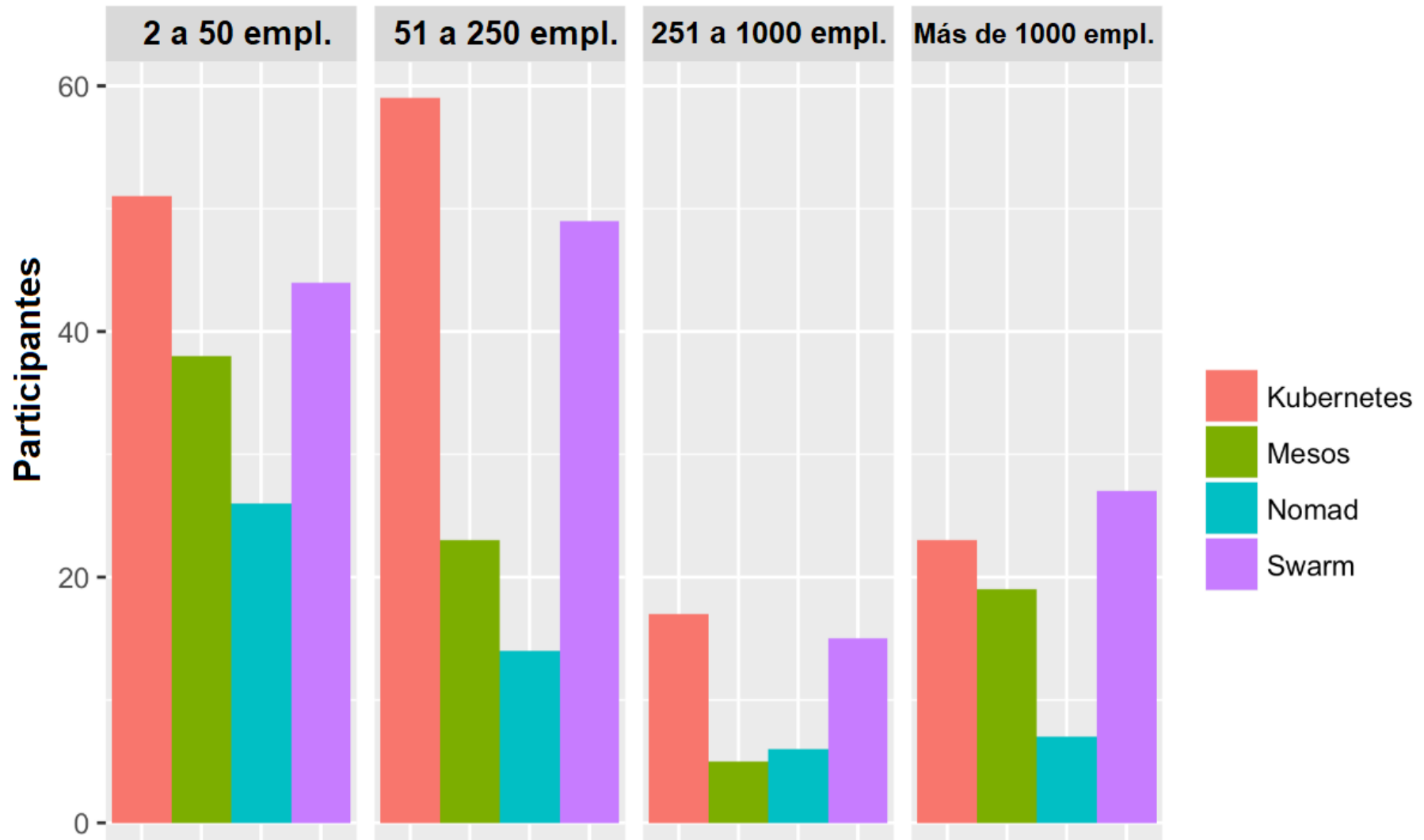
Alcance de las pruebas de servicio



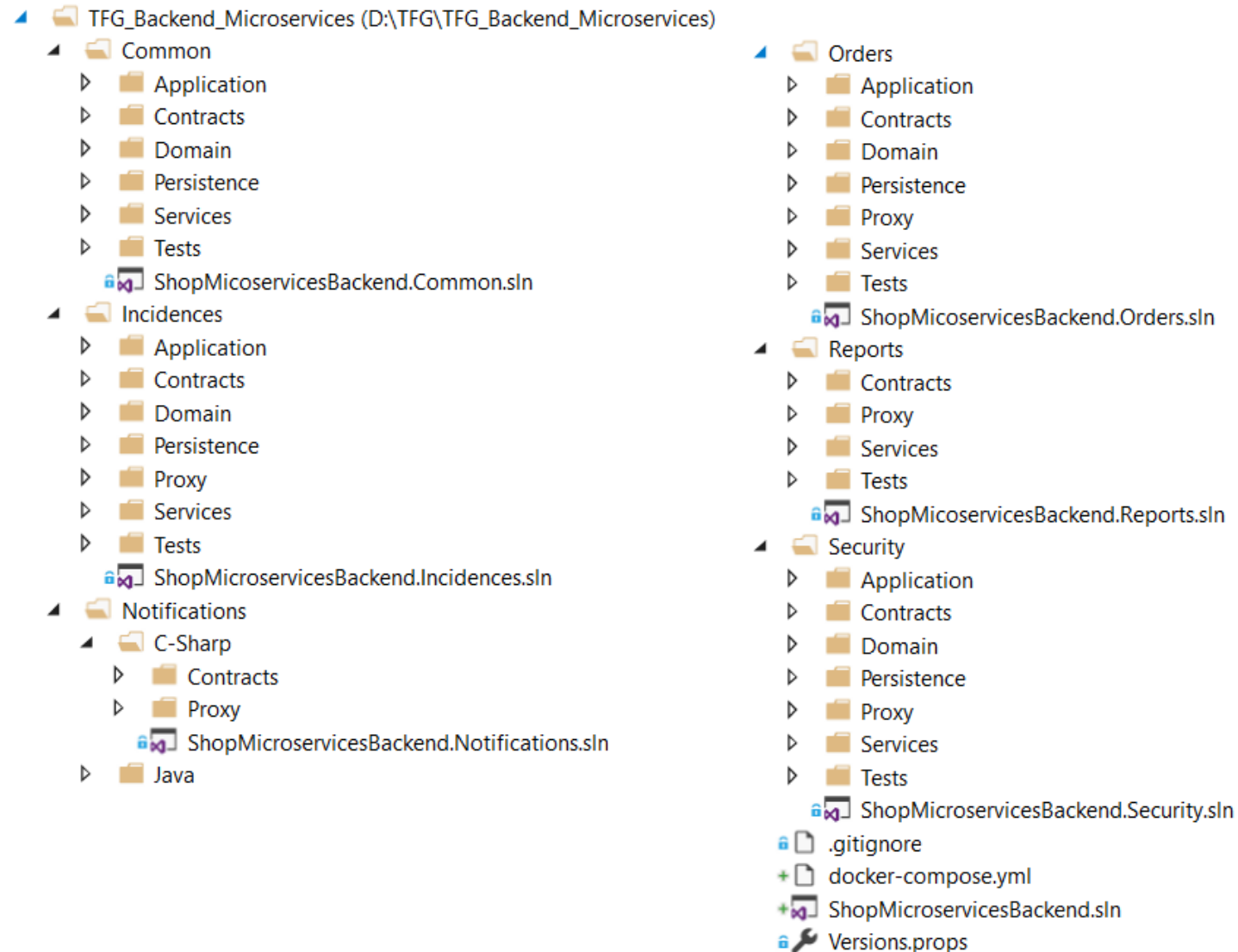
Alcance de las pruebas unitarias



Comparación de orquestadores



Solución en VS del sistema de microservicios



Plantilla de Open-Xml PowerTools

Tienda Online



Factura

Factura

Cliente: <# <Content Select= "/>Client" /> #>

State of the order: <# <Content Select= "/>State" /> #>

Artículos:

Número de artículos: <# <Content Select= "/>NumberOfArticles" /> #>

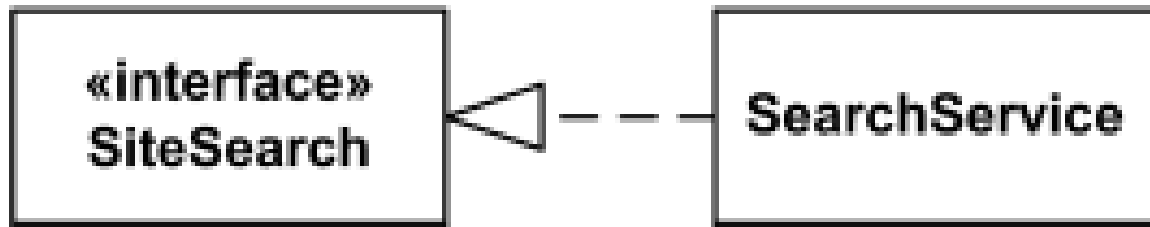
<# <Repeat Select="/>Purchases/PurchaseReportDTO"/> #>

Nombre del producto:	<# <Content Select= "/>ProductName" /> #>		
Descripción:	<# <Content Select= "/>ProductDescription" /> #>		
Nº de unidades:	<# <Content Select= "/>Quantity" /> #>	Precio unitario:	<# <Content Select= "/>Price" /> #> €
		Precio total:	<# <Content Select= "/>TotalProduct" /> #> €

<# <EndRepeat/> #>

Modelo UML

Implementación de una interfaz:



Dependencia:

Asociación unidireccional:



A2 has unspecified navigability while B2 is navigable from A2.

Capa de contratos

```
/// <summary>
/// Contracts interface for managing the orders.
/// </summary>
8 references | VictorIranzo, 65 days ago | 1 author, 7 changes
public interface IOrdersManager : ICrudManager<OrderDTO, OrderCreateDTO, OrderUpdateDTO>
{
    /// <summary> Adds the purchases and confirms the order.
    4 references | VictorIranzo, 67 days ago | 1 author, 1 change | 0 exceptions
    Task<bool> AddPurchasesAndConfirmOrder(AddPurchasesDTO addPurchasesDTO);

    /// <summary> Generates the report of and order.
    4 references | VictorIranzo, 65 days ago | 1 author, 1 change | 0 exceptions
    Task<byte[]> GenerateOrderReportAsync(Guid orderId, bool pdfFormat = true);

    /// <summary> Gets the list of user orders filtered.
    4 references | VictorIranzo, 66 days ago | 1 author, 1 change | 0 exceptions
    Task<IEnumerable<OrderDTO>> GetListOfUserOrdersFilteredAsync(Guid userId, string state = null, i
}
```

Capa de proxy

```
/// <inheritdoc/>
```

4 references | VictorIrranzo, 59 days ago | 1 author, 1 change | 0 exceptions

```
public async Task<IEnumerable<OrderDTO>> GetListOfUserOrdersFilteredAsync(Guid userId, string state = null, int fromElement)
{
    return await this.ordersClient.GetListOfUserOrdersFilteredAsync(userId.ToString(), state, fromElement.ToString(), null);
}
```

```
/// <inheritdoc/>
```

6 references | VictorIrranzo, 59 days ago | 1 author, 1 change | 0 exceptions

```
public async Task<bool> UpdateAsync(OrderUpdateDTO updateDTO)
{
    return await this.ordersClient.UpdateAsync(updateDTO);
}
```

Capa de servicios

```
/// <summary> Gets the list of the user orders in this collection.
[HttpGet("GetUserOrdersFiltered")]
[Authorize]
[ProducesResponseType(typeof(IEnumerable<OrderDTO>), 200)]
0 references | VictorIranzo, 59 days ago | 1 author, 1 change | 0 requests | 0 exceptions
public async Task<IEnumerable<OrderDTO>> GetListOfUserOrdersFilteredAsync(
    [FromHeader] string userId,
    [FromHeader] string state = null,
    [FromHeader] string fromElement = "0",
    [FromHeader] string numberOfElements = "10")
{
    bool fromResultSuccess = int.TryParse(fromElement, out int fromElementInt);
    bool numberOfElementsSuccess = int.TryParse(numberOfElements, out int numberOfElementsInt);
    bool isValid = Guid.TryParse(userId, out Guid userGuid);

    string requesterId = this.HttpContext.User.Claims.Where(c => c.Type.Equals(ClaimTypes.NameId)

    if (!requesterId.Equals(userId.ToString()))
    {
        this.Response.StatusCode = 401;
        return null;
    }
    else
    {
        return await this.ordersManager.GetListOfUserOrdersFilteredAsync(userGuid, state, fromEl
    }
}
```

Capa de aplicación

```
/// <inheritdoc/>
```

[4 references](#) | VictorIranzo, 64 days ago | 1 author, 2 changes | 0 exceptions

```
public Task<IEnumerable<OrderDTO>> GetListOfUserOrdersFilteredAsync(Guid userId, string state = null, int f
{
    List<Expression<Func<Order, bool>>> conditionalExpressions = new List<Expression<Func<Order, bool>>>();
    conditionalExpressions.Add(o => o.User.Id.Equals(userId.ToString()));

    if (!string.IsNullOrEmpty(state))
    {
        conditionalExpressions.Add(o => o.State.Equals(Order.GetOrderState(state)));
    }

    IEnumerable<Order> orders = this.EntityCrudDAO.Get(conditionalExpressions, null, null, fromElement, num
    orders.ToList().ForEach(o => o.Purchases = null);

    return Task.FromResult((IEnumerable<OrderDTO>)this.EntityDTOConverter.ConvertFromEntityToDto(orders));
}
```

Capa de dominio

```
public class Order : Entity
{
    /// <summary> Gets or sets the creation date. </summary>
    /// <value> The creation date. </value>
    2 references | VictorIrranzo, 65 days ago | 1 author, 1 change | 0 exceptions
    public DateTime CreationDate { get; set; }

    /// <summary>
    /// Gets or sets the collection of purchases of the order.
    /// </summary>
    7 references | VictorIrranzo, 86 days ago | 1 author, 2 changes | 0 exceptions
    public ICollection<Purchase> Purchases { get; set; }

    /// <summary>
    /// Gets or sets the state of the order.
    /// </summary>
    8 references | VictorIrranzo, 87 days ago | 1 author, 1 change | 0 exceptions
    public OrderState State { get; set; }

    /// <summary>
    /// Gets or sets the client that makes the order.
    /// </summary>
    15 references | VictorIrranzo, 86 days ago | 1 author, 1 change | 0 exceptions
    public User User { get; set; }
```

Capa de persistencia

```
/// <inheritdoc>
```

```
6 references | VictorIranzo, 71 days ago | 1 author, 2 changes | 0 exceptions
```

```
public IEnumerable<TEntity> Get(  
    IEnumerable<Expression<Func<TEntity, bool>>> conditionalExpressions = null,  
    Expression<Func<TEntity, object>> orderExpression = null,  
    IEnumerable<Expression<Func<TEntity, object>>> includeExpressions = null,  
    int fromElement = 0,  
    int numberOfElements = 10)  
{  
    try  
    {  
        IQueryable<TEntity> query = this.GetDbSet().AsQueryable();  
  
        if (conditionalExpressions != null)  
        {  
            foreach (Expression<Func<TEntity, bool>> conditionExpression in conditionalExpressions)  
            {  
                query = query.Where(conditionExpression);  
            }  
        }  
    }  
}
```