



Portal para comunidades de propietarios

José Luis Zorita Gutiérrez

Grado de Ingeniería Informática

ÍNDICE

- 3. Contexto
- 4. Objetivos del trabajo
- 5. Usuarios y requisitos
- 6. Prototipo
- 7. Arquitectura
- 8. Tecnologías utilizadas
- 9. Entidades Microservicios
- 10. Comunicación Kafka
- 11. API REST
- 12. Arquitectura hexagonal
- 13. Etapas implementación
- 14. Esqueleto
- 15. Infraestructura
- 16. API REST y WEB Microservicios
- 17. API REST y WEB Aplicación web
- 18. Seguridad
- 19. Problemas y dificultades
- 20. Conclusiones

Contexto

Contexto bajo el que se desarrolla el proyecto

Existen aplicaciones para que los propietarios puedan consultar información, pero suelen contar con varias deficiencias, como por ejemplo:

- Las aplicaciones actuales muestran **información muy limitada**.
- El administrador **no utiliza estas plataformas, trabaja desde su ERP**.
- **Falta de interacción entre el administrador y el propietario**.
- Los propietarios siguen haciendo sus **gestiones de forma presencial, por vía telefónica o carta**.

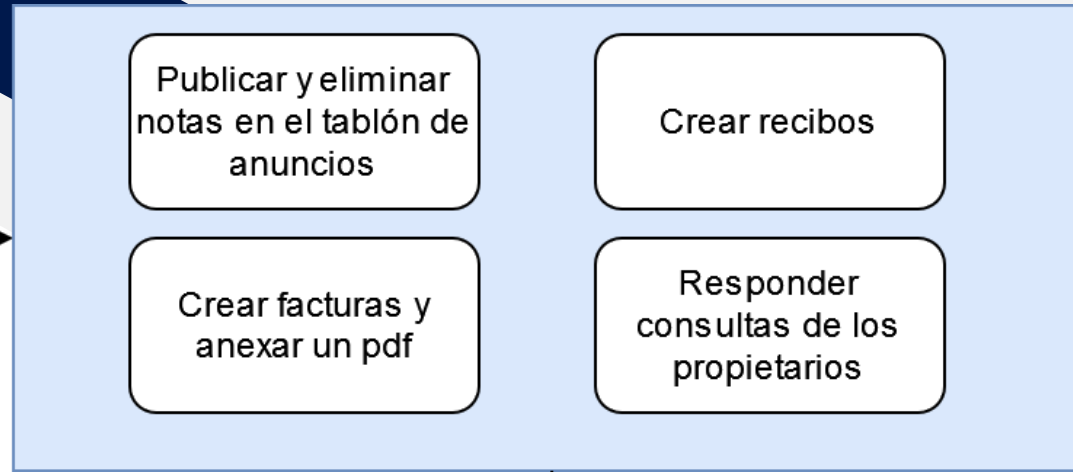
Objetivos del trabajo

El proyecto tiene los siguientes objetivos

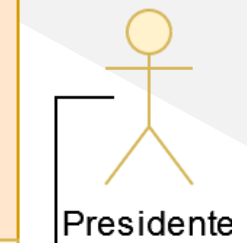
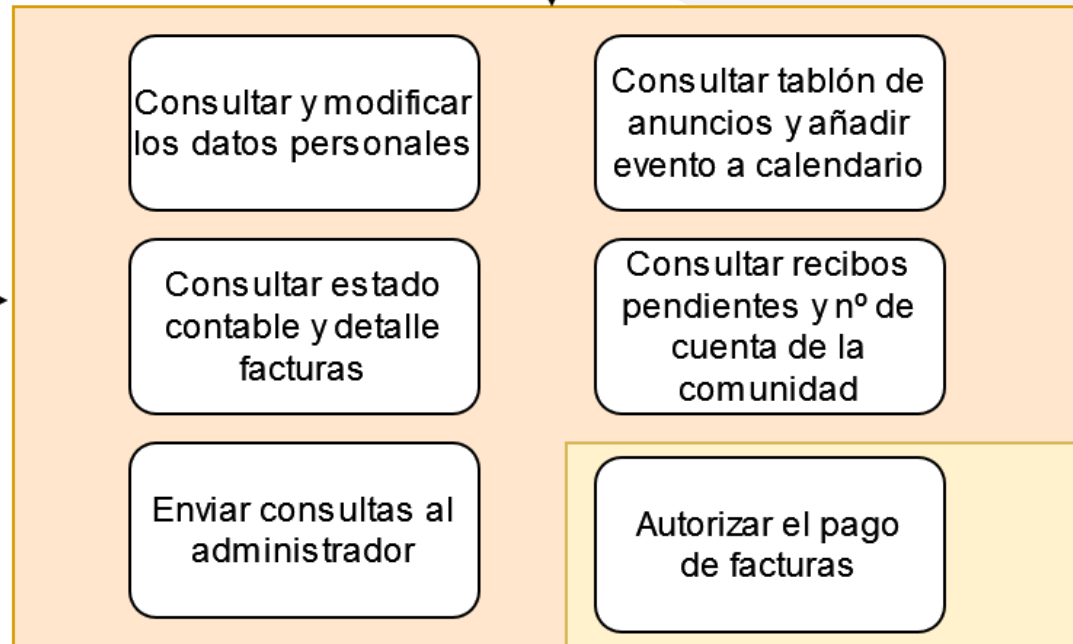
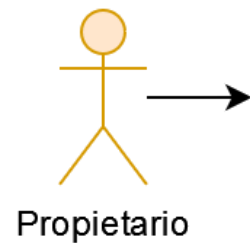
Crear una plataforma mediante microservicios que ofrezca servicios API REST a consumir desde un cliente web que permita:

- Que los **vecinos** y el **administrador** estén en el **mismo ecosistema**.
- Crear un **canal de comunicación** entre los propietarios y el administrador.
- Al propietario realizar acciones como **consultar el estado contable, autorizar facturas o cambiar sus datos**.
- Al administrador realizar acciones como **crear facturas y recibos**.

Usuarios y requisitos

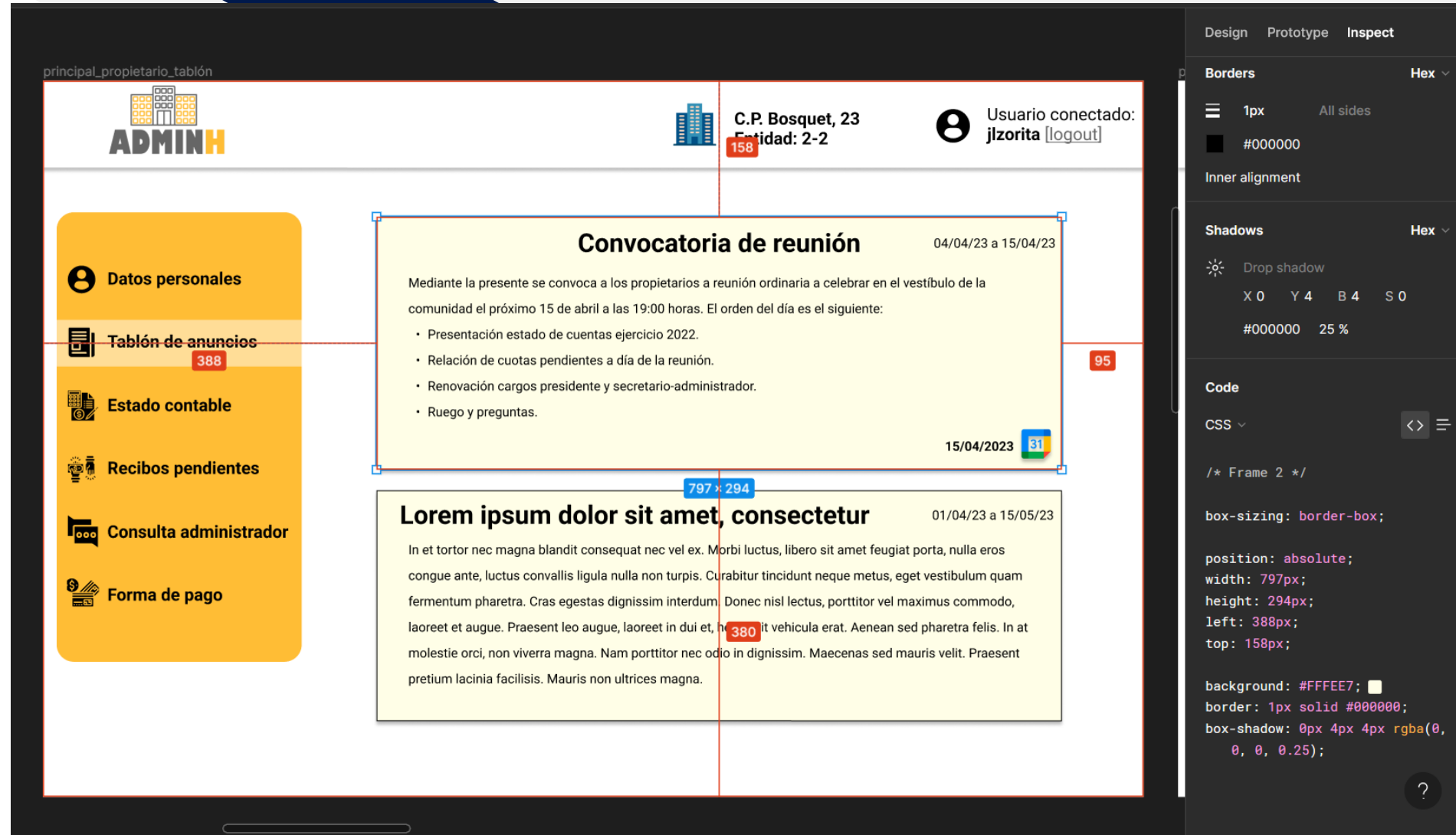


Enviar notificación eventos



Prototipo

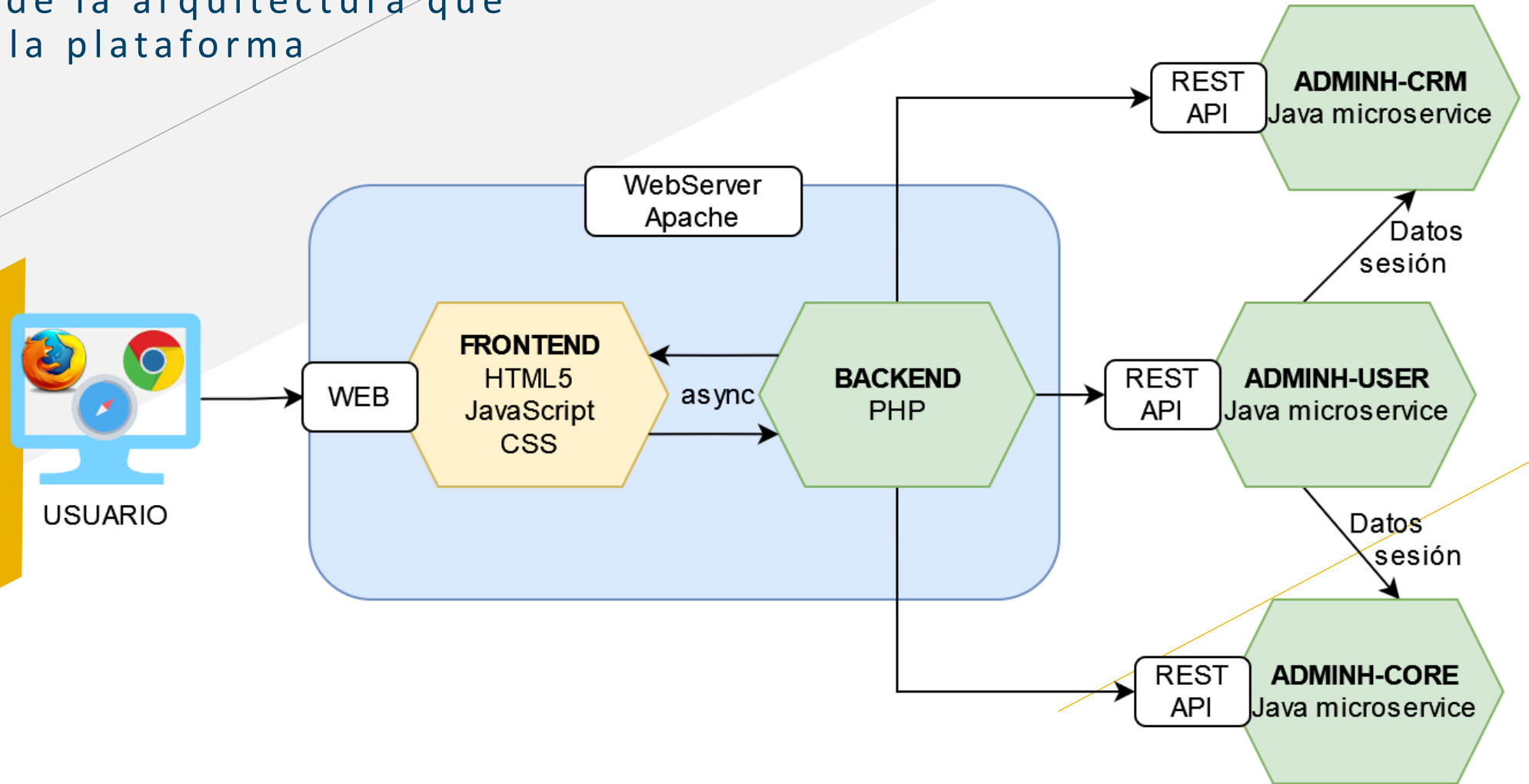
Se ha creado un prototipo con **FIGMA** que permite inspeccionar el estilo css para trasladarlo a la aplicación final



[Enlace prototipo](#)




Arquitectura

Diagrama de la arquitectura que albergará la plataforma



Detalle tecnologías utilizadas



Microservicios

Propósito	Aplicación
Lenguaje / kit de desarrollo	
Gestión del proyecto	
Persistencia / manipulación datos	
Base de datos	
Comunicación entre microservicios	
Automatización proyecto	
Librería anotaciones	
Inicialización base de datos	

Aplicación web

Propósito	Aplicación
Paquete que integra servidor apache y php	XAMPP 
Lenguajes para codificación de frontend	 
Lenguaje de programación lado cliente	Javascript 
Programación lado servidor	
Comunicación microservicios	

Entornos de desarrollo

Propósito	Aplicación
Codificación aplicación web y ficheros Docker	Visual Studio Code 
Codificación microservicios	IntelliJ 

Entidades Microservicios

Microservicios

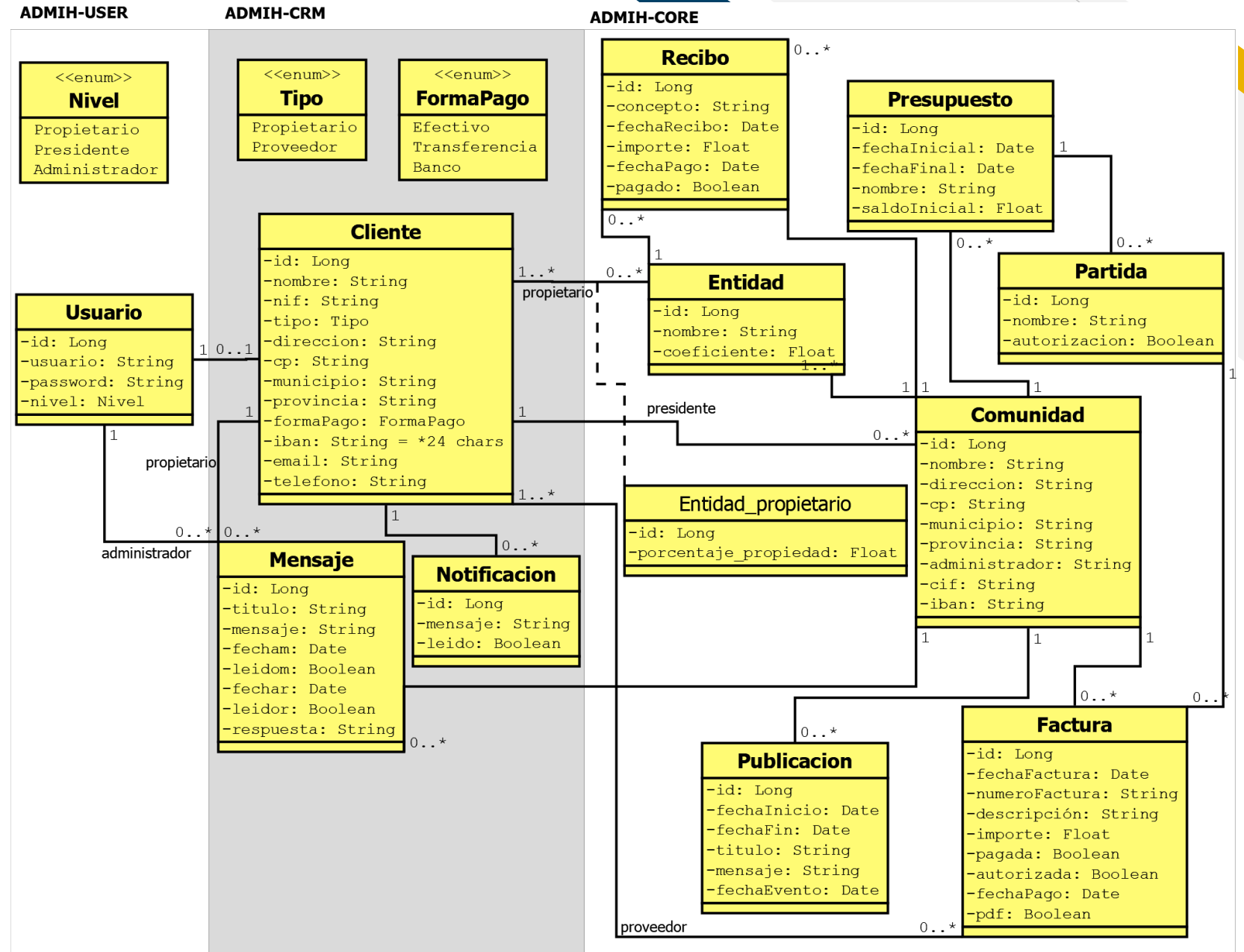
Entidades de los microservicios que conforman las tablas de sus bases de datos

Adminh-user: **db-user**

Adminh-crm: **db-crm**

adminh-core: **db-core**

Implementado en etapa 2



Comunicación Kafka

Diagrama comunicación entre microservicios

ADMINH-USER

Sesion
-sesiones: Map<String,String[]>
+addUsuario(usuario:String,sesion:String[])
+removeUsuario(usuario:String)
+comprobarSesionExiste(usuario:String): Boolean
+getSession(usuario:String): String[]
+setLevel(usuario:String,nivel:Integer,clienteId:Long): boolean
+enviarSesion(sesionData:SesionData,KafkaTemplate:KafkaTemplate): Long

SesionData
-usuario: String
-sesion: String[3]
-alta: Boolean

SesionData.sesion:
{usuario, nivel, clienteId}

KAFKA PRODUCER

ProducerFactory<String, SesionData>
KafkaTemplate<String, SesionData>

Topics

KafkaConstants.TOPIC_SESSION_CRM
KafkaConstants.TOPIC_SESSION_CORE

Enviar sesion : método enviarSesion(sesionData, kafkaTemplate)

```
kafkaTemplate.send(KafkaConstants.TOPIC_SESSION_CRM, sesionData);  
kafkaTemplate.send(KafkaConstants.TOPIC_SESSION_CORE, sesionData);
```

ADMINH-CRM

Cliente
... datos cliente ...
-sesiones: Map<String,String[]>
+addUsuario(usuario:String,sesion:String[])
+removeUsuario(usuario:String)
+comprobarNivelUsuario(sesion:String): String
+getSession(usuario:String): String[]
+setSesion(sesion:SesionData)

SesionData
-usuario: String
-sesion: String[3]
-alta: Boolean

TOPIC_SESSION = "sessions.CRM"

KAFKA CONSUMER

KAFKA ClassListener
(TOPIC_SESSION, groupId = "group-1")
setSesion(sesionData sesion);

ADMINH-CORE

Cliente
-sesiones: Map<String,String[]>
+addUsuario(usuario:String,sesion:String[])
+removeUsuario(usuario:String)
+comprobarNivelUsuario(sesion:String): String
+getSession(usuario:String): String[]
+setSesion(sesion:SesionData)
+getCliente(sesion:String): String

SesionData
-usuario: String
-sesion: String[3]
-alta: Boolean

TOPIC_SESSION = "sessions.CORE"

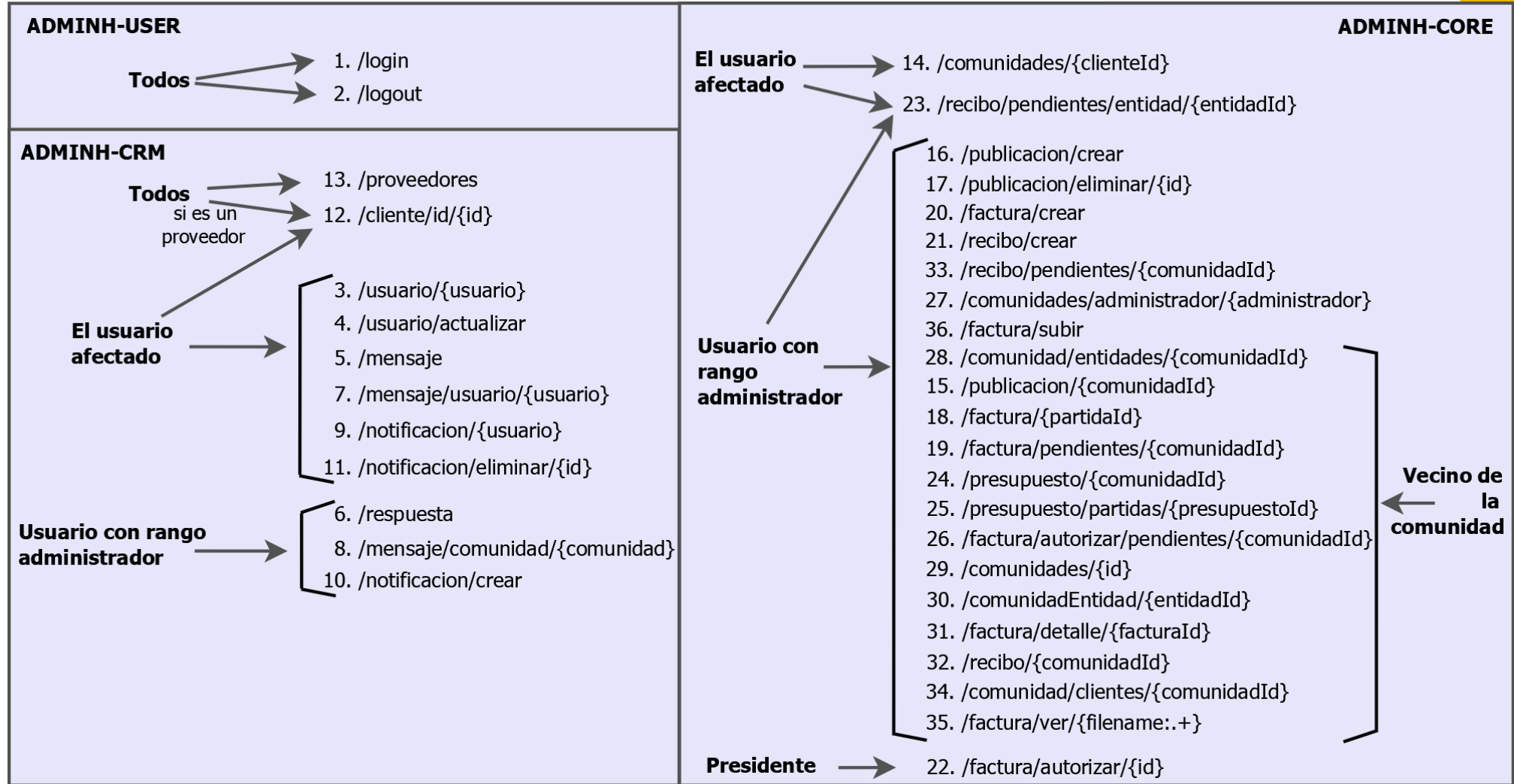
KAFKA CONSUMER

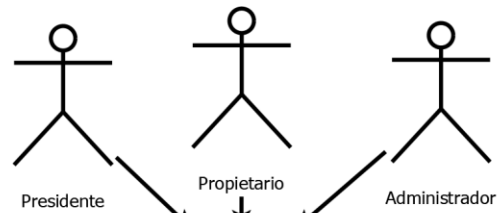
KAFKA ClassListener
(TOPIC_SESSION, groupId = "group-1")
setSesion(sesionData sesion);

API REST

Servicios API REST
Microservicios

Métodos de los servicios API REST y permisos de usuario para su ejecución





```

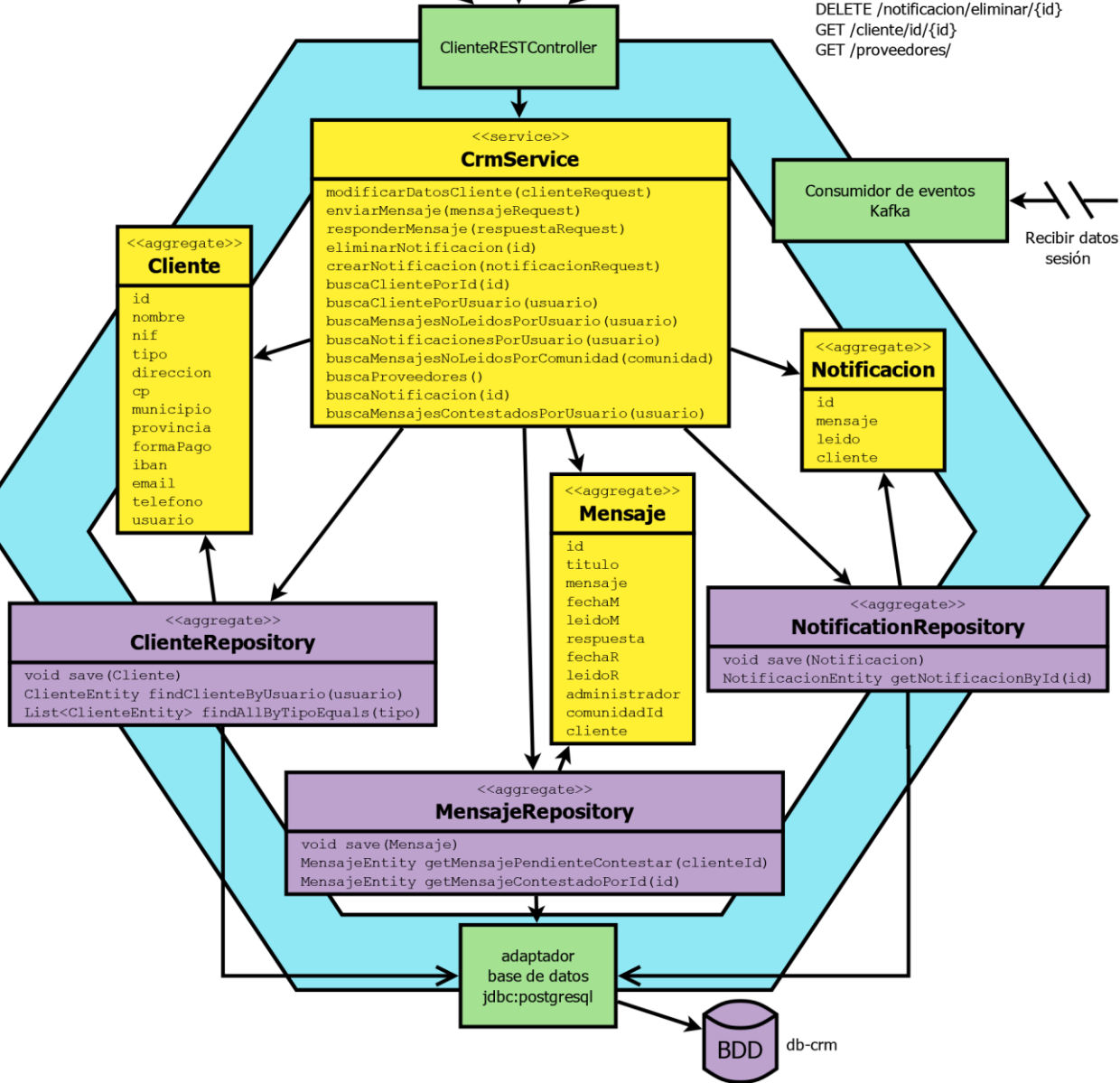
GET /user/sesion/{usuario}
GET /usuario/{usuario}
PUT /usuario/actualizar
POST /mensaje
PUT /respuesta
GET /mensaje/usuario/{usuario}
GET /mensaje/comunidad/{comunidadId}
GET /notificacion/{usuario}
POST /notificacion/crear/
DELETE /notificacion/eliminar/{id}
GET /cliente/id/{id}
GET /proveedores/

```

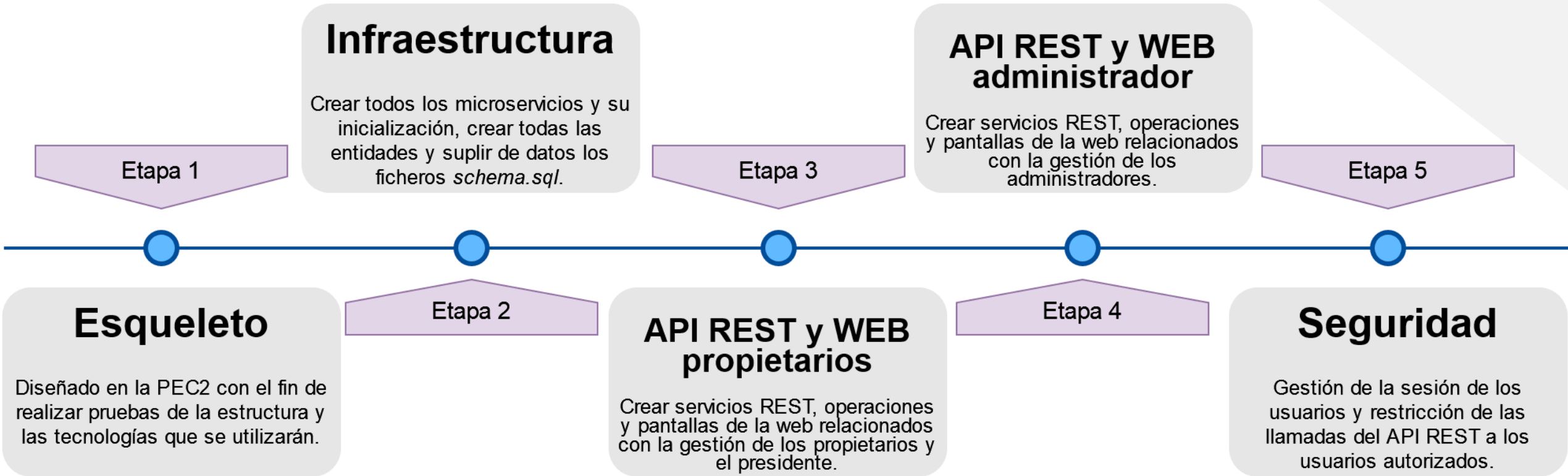
Arquitectura hexagonal

Arquitectura hexagonal
adminh-crm

Ejemplo de la arquitectura
seguida por el
microservicio adminh-crm.



Etapas implementación



Esqueleto

Etapa 1 de la implementación

```
[+] Running 7/7
- Container tfg-db-user-1      Healthy
- Container tfg-zookeeper-1   Started
- Container tfg-adminer-1     Started
- Container tfg-db-crm-1      Started
- Container tfg-kafka-1       Healthy
- Container tfg-adminh-user-1 Started
- Container tfg-adminh-crm-1  Started
PS C:\TFG>
```

*Arranque microservicios
con docker-compose*

Creado un esqueleto de forma previa a la PEC 3 con el objetivo de:

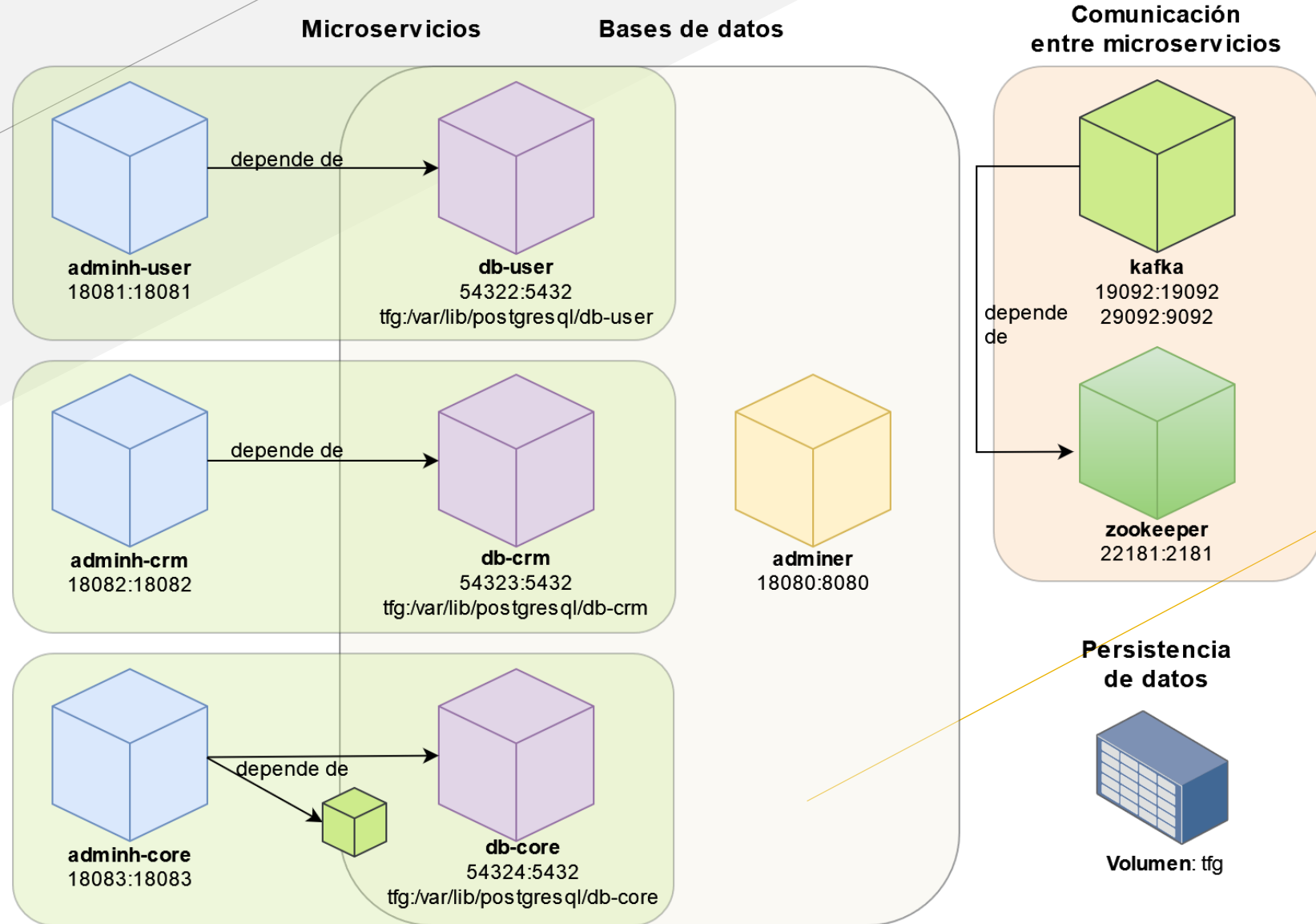
- Realizar una **configuración inicial del entorno.**
- **Comparar posibles tecnologías** a utilizar para su aprobación final.
- **Probar las herramientas** que se utilizarán en el posterior desarrollo.
- **Dejar el sistema preparado** para iniciar la fase de desarrollo.

Infraestructura

Etapa 2 de la implementación

Objetivos de esta etapa:

- Acabar de **configurar toda la infraestructura**.
- **Crear las entidades** en cada microservicio según el esquema de entidades.
- **Suplir a los ficheros *schema.sql* de información** para inicializar con datos la aplicación.

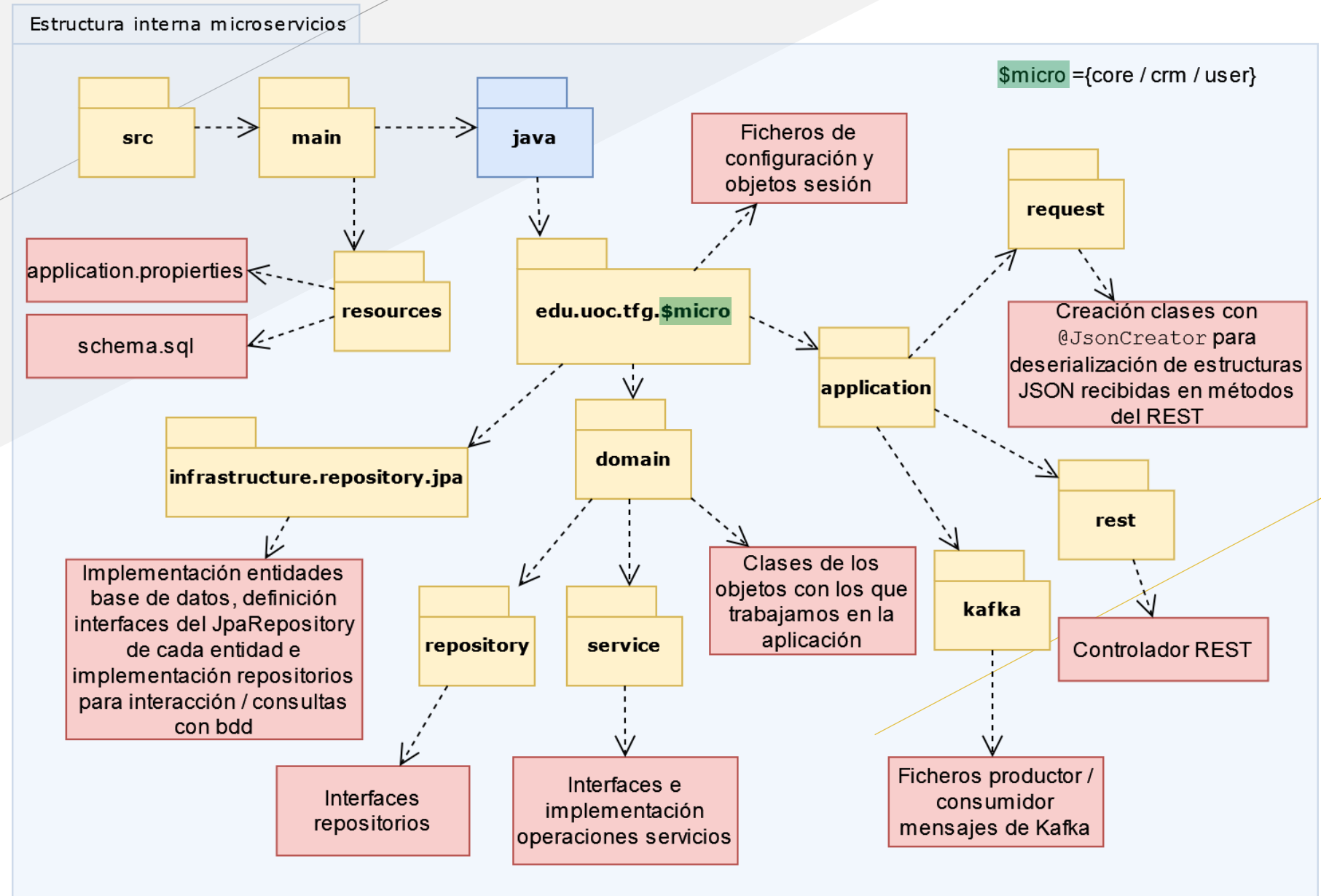


Servicios definidos en `docker-compose.yml`

API REST y WEB - Microservicios

Etapas 3 y 4 de la implementación

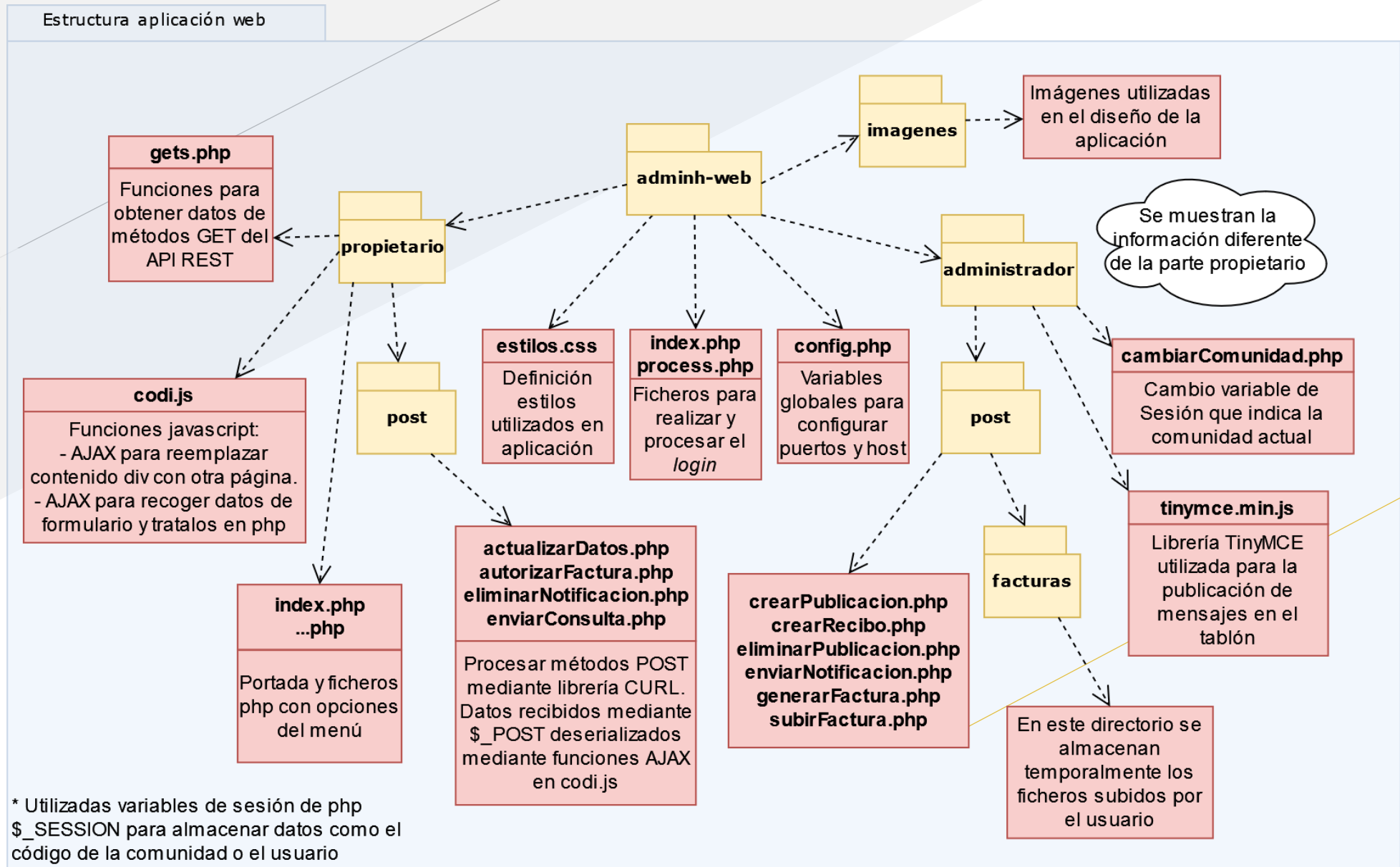
Se muestra en rojo el detalle de la codificación realizada



API REST y WEB - Aplicación web

Etapas 3 y 4 de la implementación

Se muestra en rojo el detalle de la codificación realizada



Seguridad

Etapa 5 de la implementación

- Modificación de todos los **métodos del API REST** para **verificar el código de sesión** y que **sólo puedan ser ejecutados por usuarios autorizados**.
- Ha supuesto la necesidad de **crear nuevas operaciones para realizar verificaciones**.
- **Código 401 UNAUTHORIZED** en caso de no contar con permisos.

The screenshot shows a REST client interface for a request to `http://localhost:18082/mensaje/comunidad/1?sesion=CODIGOINCORRECTO`. The response is `401 Unauthorized` with a status of `8 ms` and `222 B`. The response body is currently empty.

Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	sesion	CODIGOINCORRECTO		
Key	Value	Description		

Problemas y dificultades

Dificultades encontradas durante la implementación de la plataforma

- **Error de compilación** al producirse un bucle infinito por relación entrelazada entre entidades al obtener datos [`toDomain()`].
- Error a la hora de **insertar datos** si en las instrucciones INSERT INTO del `schema.sql` se especificaba manualmente un identificador.
- En aplicación web error por **falta de permisos** de escritura en directorio en el que se suben temporalmente los ficheros.

Conclusiones

- ✓ **Valoración global positiva:** se han alcanzado objetivos y la aplicación se asemeja a prototipo de Figma.
- ✓ **Infraestructura configurada antes de la fase de la implementación** debido a la configuración de un esqueleto de forma previa.
- ✓ **Profundización** en diferentes tecnologías para el desarrollo de microservicios.
- ✓ **Aprendizaje de nuevas herramientas** y lenguajes para desarrollo web.
- ✗ **Gran desviación respecto a la planificación inicial.**
- ✗ **Falta de pruebas en el código.**
- ✗ **No se han utilizado *frameworks* para la aplicación web.**