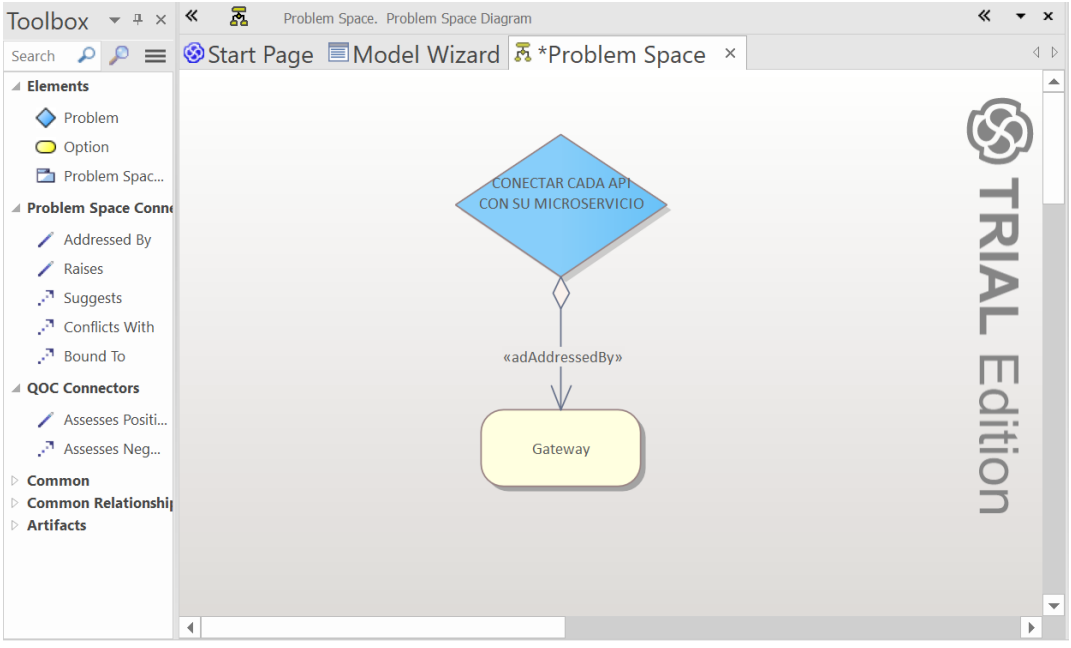
SEMANA 4: ITERACIÓN 3 DE LA ARQUITECTURA (ASJ)

Una vez obtenido todas las decisiones, los Juniors empezamos con la realización de diseño de los diagramas UML en base a las capturas de las decisiones de diseño aceptadas.

En primer lugar, se ha tomado la decisión ADD-08 “Uso de un Gateway a diferentes APIs” mediante el uso de un Gateway, correspondiente a la necesidad que se explica en el requisito RF-001.2 (consulta de actualizaciones) y el requisito RF001.3 (Comunicación de clientes con microservicios). Consecuentemente tal decisión engloba una serie de ventajas (Escalabilidad, modularidad, bajo acoplamiento, capa extra de seguridad y posibilidad de añadir funciones intermedias entre cliente y servicio) y desventaja (mayor complejidad en arquitectura y código).

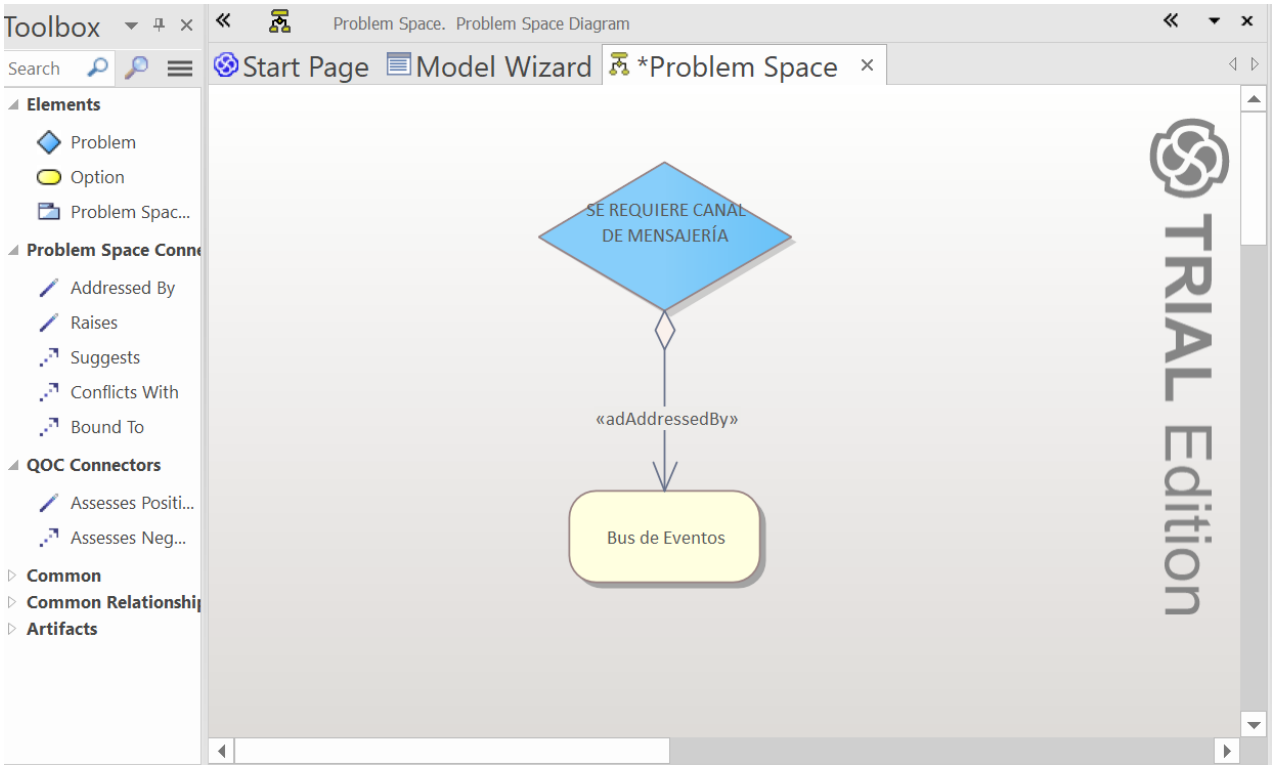
```
# Decision de diseño ADD-08: Uso de un gateway a diferentes APIs
* Status: proposed <!-- optional -->
* Deciders: Antonio Agudo Esperanza y Marcos Robles Rodríguez<!-- optional -->
* Date: 2020-12-1 <!-- optional -->

## Context and Problem Statement

El cliente nos ha expuesto que usemos un gateway y hemos decidido conectarlo a unas API cada una con su microservicio para que sea posible una modularización
óptima

## Decision Drivers <!-- optional -->

* El cliente comenta el uso de una gateway
* Busqueda de completitud a la hora de comunicar cliente servidor
* Busqueda de la máxima modularidad por el uso de una API para cada microservicio

## Considered Options

* Uso de gateway
* Conexion directa entre el cliente con los microservicios

## Decision Outcome

Chosen option: Uso de el gateway, porque consideramos muy importante la modularización y el bajo acoplamiento para que sea posible una mejor escalabilidad en
nuestra arquitectura, además, las APIs y el gateway puede realizar acciones intermedias y puede servir como una capa extra de seguridad.

### Positive Consequences <!-- optional -->

* Escalabilidad
* Modularidad
* Bajo acoplamiento
* Capa extra de seguridad
* Posibilidad de añadir funciones intermedias entre el cliente y el servicio

### Negative Consequences <!-- optional -->

* Mayor complejidad en la arquitectura y el código.

## Pros and Cons of the Options <!-- optional -->

### Cliente servidor directo>

* Bueno, ya que es más simple de programar y hacer.
* Malo, ya que el cliente debe conocer a la perfección los microservicios ya que no hay ninguna API que se los facilite
* Malo, porque no hay nada que se interponga entre una mala solicitud y los microservicios dejando un agujero de seguridad

## Links <!-- optional -->

* [Requisito Funcional 1.2 y 1.3](https://github.com/Grupo3-DAS/Pr-ctica1-Captura-y-Representaci-n-de-Decisiones-de-Dise-o-Equipo-3/blob/main/DAS-P1-
Alba_Sevillano_Portilla-TAREA1.pdf)
```



Adicionalmente, en esta misma iteración, se ha considerado la decisión ADD-09 “Canal de mensajería mediante un bus de eventos” tras observar el problema que se explica en el requisito funcional RF-002 (canal de mensajería) y el requisito funcional RF-002.1 (comunicar los servicios con una API REST y de forma asíncrona).

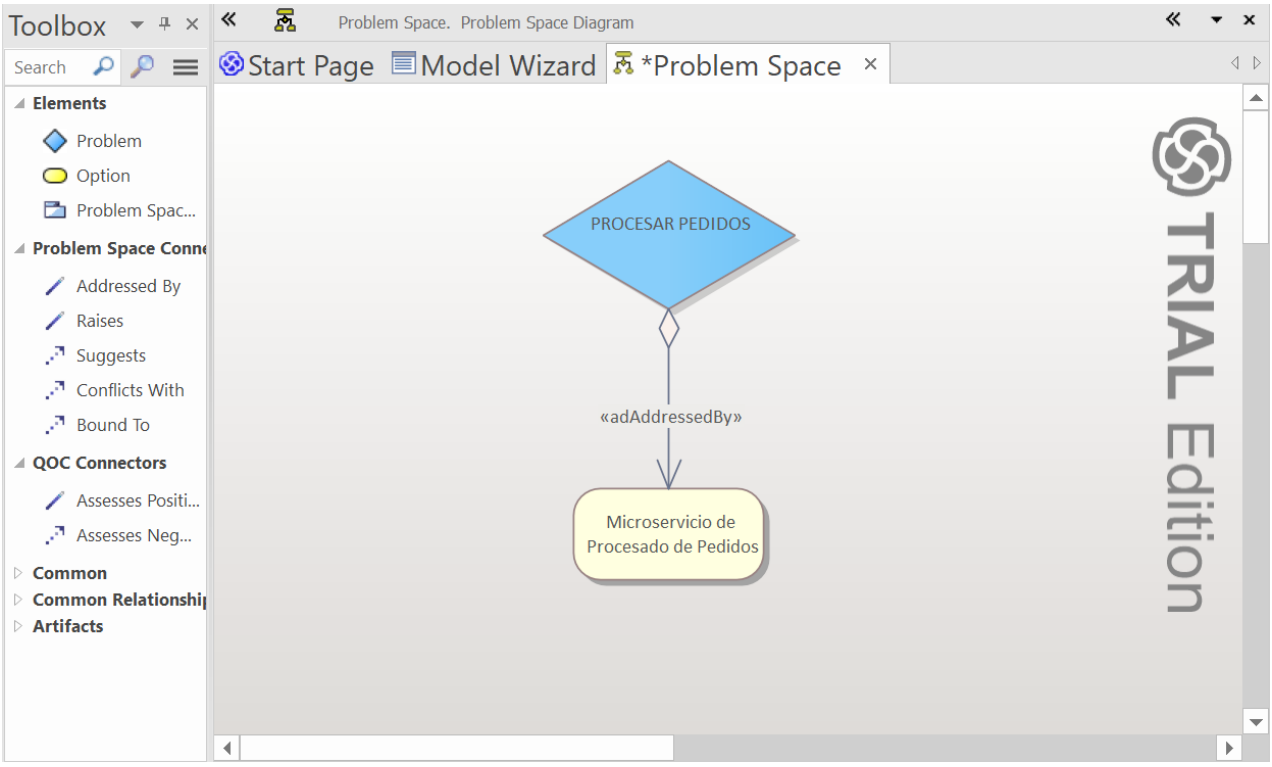
Consecuentemente le siguen a tal decisión una serie de ventajas (Permite procesar un mismo evento por varios microservicios, se desvincula cada microservicio de los eventos, hay un sistema de comunicación general que facilita añadir nuevos microservicios y alta escalabilidad) e inconveniente (se tiene que agregar un subsistema para garantizar la entrega del evento).

```
# Decisión de Diseño ADD-09: Canal de mensajería mediante un event bus
* Status: proposed
* Deciders: Antonio Agudo Esperanza
* Date: 2020-12-01
## Context and Problem Statement
* La lógica de aplicación deberá contener un canal de mensajería para que se comuniquen los diferentes microservicios entre ellos. Este deberá estar basado
principalmente en agentes de mensajes.
## Decision Drivers
* Los microservicios deberán comunicarse entre ellos
* El cliente requiere una implementación del bus de eventos
* La mensajería debe ser asíncrona
## Considered Options
* Bus de eventos
* Publish and Subscribe
## Decision Outcome
* Chosen option: Bus de eventos ya que permite un procesamiento sencillo de eventos y permite procesar un mismo evento por varios microservicios.
### Positive Consequences
* Permite procesar un mismo evento por varios microservicios.
* Se desvincula cada microservicio de los eventos
* Hay un sistema de comunicación general que facilita añadir nuevos microservicios
* Alta escalabilidad
### Negative Consequences
* Se tiene que agregar un subsistema para garantizar la entrega del evento.
## Pros and Cons of the Options
### Publish and Subscribe

* Bueno, ya que se puede realizar una implementación propia.
* Malo, ya que hay que definir métodos específicos para cada microservicio.
* Malo, ya que puede producir ralentizaciones
* Malo, porque puede producir sobrecargas.
## Links <!-- optional -->

* [Requisito RF002 y RF002.2](https://github.com/Grupo3-DAS/Pr-ctica1-Captura-y-Representaci-n-de-Decisiones-de-Dise-o-Equipo-3/blob/main/DAS-P1-
Alba_Sevillano_Portilla-TAREA1.pdf)
```



También se ha tomado la decisión ADD-11 “Implementar el procesado de pedidos como un microservicio”, tras detectar que la aplicación requiere un profe. Esta decisión de diseño corresponde al requisito RF-001.1 (Procesamiento de pedidos por parte de la APP). Consecuentemente le siguen a tal decisión una serie de ventajas (se quita la carga del Gateway y se mantiene una arquitectura modularizada) y desventaja (puede resultar más complicado de desarrollar).

```
# Decisión de Diseño ADD-11: Implementar el procesado de pedidos como un microservicio
* Status: proposed
* Deciders: Marcos Robles Rodríguez
* Date: 2020-12-02
## Context and Problem Statement
* Se pide que el sistema sea capaz de procesar los pedidos que le llegan.
## Decision Drivers
* Respetar la arquitectura de microservicios
* Quitar carga de partes del sistema que no sean un microservicio
## Considered Options
* Hacer que el gateway procese los pedidos
* Crear un microservicio que procese los pedidos
## Decision Outcome
* Chosen option: Implementaremos el nuevo microservicio por separado ya que esto nos permitirá seguir con la modularidad y así no le ponemos tareas
adicionales al gateway
### Positive Consequences
* Se quita carga del gateway
* Se mantiene una arquitectura modularizada
### Negative Consequences
* Puede resultar más complicado de desarrollar
## Pros and Cons of the Options
### Hacer que el gateway procese los pedidos
Se pensaba que según le entraban las solicitudes al gateway este comprobara si era un pedido y lo procesara él.
* Bueno, ya que es más sencillo de desarrollar
* Malo, ya que se está añadiendo funcionalidades que no corresponden al gateway
* Malo, ya que si hay mucho pedidos el gateway podría retrasarse en sus otras tareas
## Links <!-- optional -->

* [Requisito funcional 001.1](https://github.com/Grupo3-DAS/Pr-ctica1-Captura-y-Representaci-n-de-Decisiones-de-Dise-o-Equipo-3/blob/main/DAS-P1-
Alba_Sevillano_Portilla-TAREA1.pdf)
```



Por último, se ha tomado la decisión ADD-012 “Elección de contenedor de nuestros microservicios”, tras detectar que la aplicación requiere del uso de contenedores para cada microservicio. Esta decisión de diseño corresponde al requisito RF-002.1.1 (Uso de contenedores de microservicios para orquestar contenedores). Consecuentemente le siguen a tal decisión una serie de ventajas (Mejora continua, implementación rápida, capacidad de restaurar a una versión anterior del microservicio y modularidad) y desventaja (problemas de seguridad).

```
# Decisión de Diseño ADD-12: Elección de contenedor de nuestros microservicios
# Elección de contenedor de nuestros microservicios, con Docker
* Status: proposed
* Deciders: Antonio Agudo Esperanza y Marcos Robles Rodriguez
* Date: 2020-12-02
## Context and Problem Statement
* Necesitamos usar un contenedor para guardar cada uno de los microservicios y todo lo necesario para la parte del servidor de nuestra arquitectura
# Decision Drivers
* Que los contenedores estén modularizados
* Escalabilidad

## Considered Options
* Usar docker host

## Decision Outcome
* Chosen option: Docker ya que permite buena escalabilidad y está preparado para ejecutarse rápidamente

### Positive Consequences
* Mejora continua
* Implementación rápida
* Capacidad de restaurar a una versión anterior del microservicio
* Modularidad

### Negative Consequences
* Problemas de seguridad

## Links <!-- optional -->

*  [Requisito funcional 002.1.1](https://github.com/Grupo3-DAS/Pr-ctica1-Captura-y-Representaci-n-de-Decisiones-de-Dise-o-Equipo-3/blob/main/DAS-P1-Alba_Sevillano_Portilla-TAREA1.pdf)
```