

USO DE SOLUCIONES CLOUD PARA LA AUTOMATIZACIÓN DE PROCESOS

28/06/2020

Álex Sánchez Ortega

Contenido

1.	Introducción	4
2.	Objetivo	5
3.	Estructura del proyecto	6
3.1	Plataforma Web	6
3.2	Desarrollo API.....	7
3.3	Automatización de procesos con Logic Apps	7
3.4	Monitorización de errores	11
4.	Metodología	12
5.	Desarrollo	12
5.1	Toma de requisitos	12
5.2	Diseño	25
5.3	Creación de la plataforma web.....	34
5.4	Configuración de la base de datos.....	36
5.5	Configuración de la API	39
5.6	Deploy	52
5.7	Logic Apps	55
6.	Presupuesto	70
7.	Planificación.....	72
7.1	Inicio del Proyecto	72
7.2	Captación de requisitos	73
7.3	Preparación del entorno y diseño.....	73
7.4	Desarrollo y Test	73

7.5	Test y rendimiento	74
7.6	Reorganización de la planificación.....	74
7.7	Trello	75
8.	Conclusiones.....	76
8.1	Puntos para mejorar	78
9.	Bibliografía.....	79
10.	Figuras	82
11.	Tablas.....	84
11.1	Tablas de requisitos funcionales	84
11.2	Tablas de requisitos de datos	84
11.3	Tablas de requisitos de datos	85

1. Introducción

La gestión de registros es una de las funciones imprescindibles en cualquier empresa u organización. A parte de ser obligatoria, da la posibilidad de tener un control exhaustivo de cualquier movimiento y acción dentro de la entidad. Este proceso aborda tres fases del ciclo de vida de los registros: la creación o recepción; el mantenimiento, el almacenamiento, la recuperación o el uso general; y la disposición. Sabiendo que en los registros reside todo lo que sucede en la empresa y son de gran importancia de cara a las entidades legislativas, los clientes y la gestión de la propia empresa, hay que asegurar un almacenamiento seguro de éstos.

Hoy en día, una empresa puede almacenar cientos de miles, o incluso millones, de registros y, más allá de las preocupaciones legales, debe haber un propósito adicional para su almacenamiento. Es decir, debe haber un enfoque que beneficie a la propia empresa y/o al cliente de alguna manera.

Partiendo de este punto, el proyecto está relacionado con dar una disposición a cualquier tipo de dato de los registros almacenados por una empresa que trabaja en la cadena de suministros. Desde el etiquetado de productos, la gestión de envíos, llegadas, facturas de pedidos, etc.

Antes de nada, para trabajar con la gestión de registros y dar un acceso a estos se debe concretar de donde salen éstos y cómo van a ser registrados.

Por otro lado, para tener un registro de estos datos, por parte de los usuarios, debe haber una herramienta que dé la posibilidad de registrarlos. En una cadena de suministros son varios los métodos para ejercer esta función, como por ejemplo, mediante registros físicos o, como es más común, con dispositivos móviles o plataformas web. Para éstas dos últimas, el sistema de almacenaje suele ser común, es decir, en bases de datos accedidas desde ambos dispositivos.

Para acabar, hay que señalar que estos sistemas, capaces de registrar cuantiosas cantidades de datos, deben tener una monitorización y un control del flujo de las acciones. Esto es lo que permitirá dar fiabilidad y consistencia a los datos registrados por los usuarios y evitar cualquier tipo de error que provoque su pérdida, duplicación, etc. Según el impacto de la entidad, un error inesperado puede significar millones de datos erróneos o perdidos, en cuestión de minutos.

2. Objetivo

El propósito del proyecto se divide en dos partes.

En primer lugar, crear una plataforma web que pueda mostrar los registros de datos almacenados a lo largo de una cadena de suministros.

En segundo lugar, crear flujos de trabajo automatizados que puedan tratar los datos y dar una disposición de ellos a los usuarios.

El primer punto está enfocado en dar una funcionalidad a los usuarios finales para poder gestionar los datos de sus registros de forma eficiente y personalizada.

El segundo punto busca mejorar la eficiencia de la plataforma y darle una mejor fiabilidad.

3. Estructura del proyecto

3.1 Plataforma Web

Como se ha comentado anteriormente, hay varios medios para gestionar los registros y datos que se generan a lo largo de una cadena de suministros pero, al final, la forma más eficiente de hacerlo es mediante un aplicación o software que permita personalizar la disposición de éstos.

En este caso, se opta por la realización de una plataforma web, por su accesibilidad global, simultaneidad de usuarios y la disponibilidad de la información en un formato más visual, a diferencia de un dispositivo móvil. Por otro lado, éste último debe ser más eficiente a la hora de registrar los *datos in situ*, por ejemplo, a la hora de entregar un pedido.

A la hora de hacer el desarrollo de ésta, se ha optado por .NET principalmente por su soporte para múltiples lenguajes, el fácil desarrollo basado en componentes y la compilación *just-in-time*, que permite generar el código máquina propio de la plataforma, aumentando el rendimiento de la aplicación al ser específico para cada plataforma. Éste último aspecto es muy importante, ya que necesitamos una plataforma eficiente y óptima, capaz de acceder a muchos datos lo más rápido posible.

Como aliciente, cabe señalar, como se comenta más adelante, las soluciones de Microsoft Azure son compatibles con .NET.

3.2 Desarrollo API

Para que sea posible la automatización de los procesos de la plataforma, se necesita una API personalizada que permita, abstraer funciones y ser ejecutadas desde llamadas externas.

Esta biblioteca de subrutinas debe poder realizar las funciones necesarias para que la plataforma sea eficiente y, a grandes rasgos, útil.

3.3 Automatización de procesos con Logic Apps

Como se ha mencionado anteriormente, se utilizan recursos de Microsoft Azure, un conjunto de servicios de informática en la nube que se utilizan en una extensa cantidad de empresas y entidades. Ofrece mucha flexibilidad y globalización.

Entre sus recursos disponibles se encuentra la posibilidad de crear máquinas virtuales, sistemas de almacenamiento, monitorización, migraciones, etc.

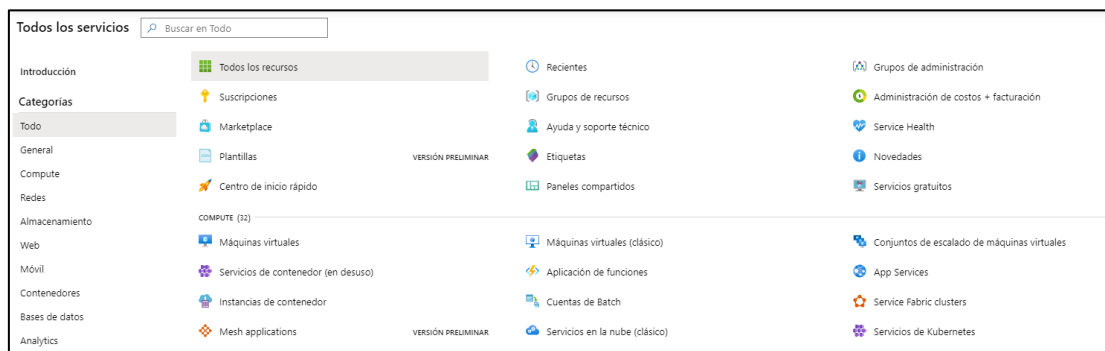


Figura 1. Microsoft Azure Services

Cabe señalar que Microsoft Azure es de pago y que para ello se debe tener una suscripción y hacer una contratación de sus servicios, normalmente de pago por uso.

En el proyecto, principalmente, se trabaja con el servicio cloud *Logic Apps*.

Éste permite automatizar y orquestar flujos de trabajo y procesos de una manera visual, organizada y simplificada.

Entre sus principales aplicaciones está la de llamar a APIs y realizar procesos mediante un desencadenador.

Por ejemplo, en el caso siguiente se aprecia un flujo de trabajo de ejemplo realizado para probar este tipo de servicio.

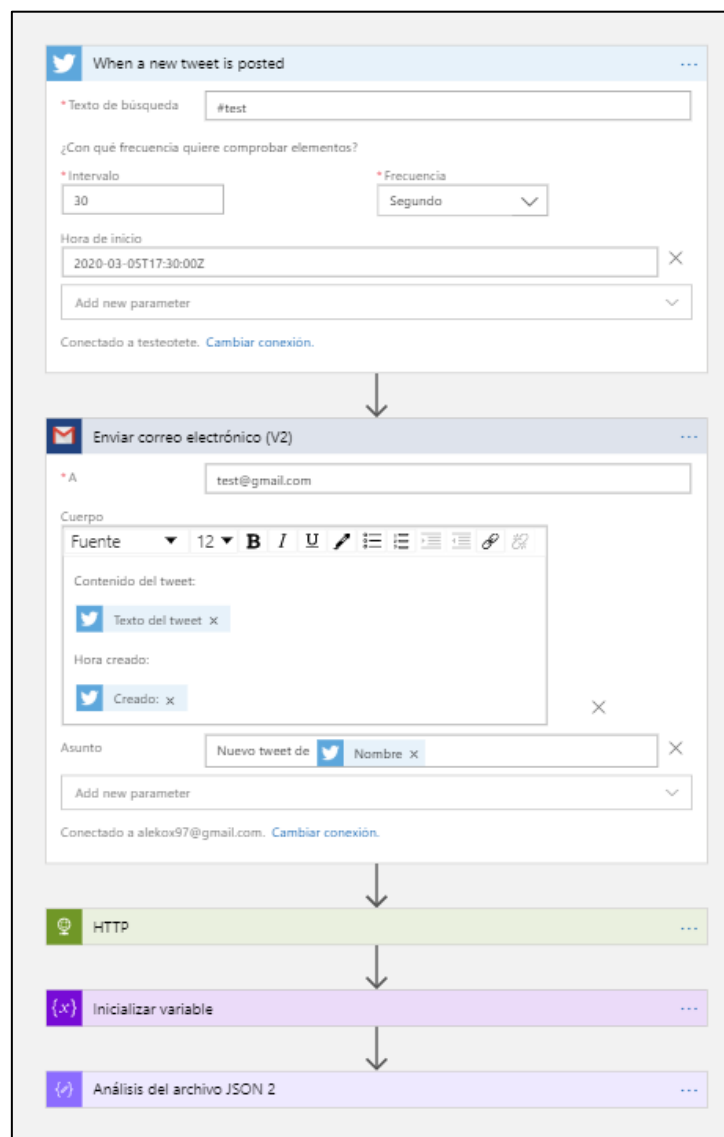


Figura 2. Interfaz Logic Apps

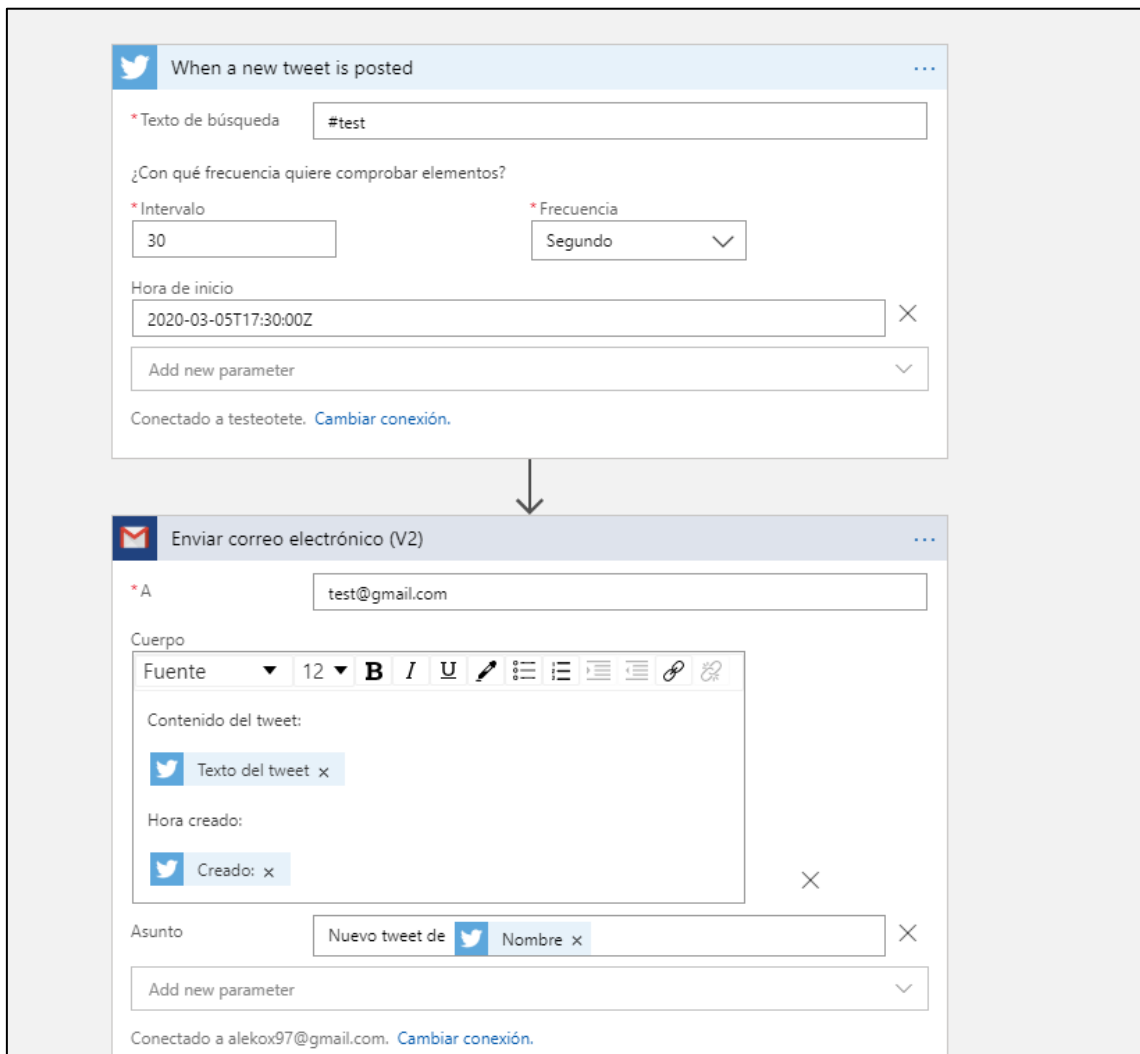


Figura 3. Flujo de trabajo de ejemplo

En la Figura 3, podemos apreciar dos cajas, donde cada una de ellas expresa una acción.

La primera de ellas es el desencadenador de la aplicación lógica, es decir, el **qué tiene que pasar para que se active**.

La segunda, es la acción que encadena el desencadenador.

En el ejemplo, se puede apreciar que (1) si se publica un tweet que contenga la etiqueta *#test* se activará la Logic App. Seguidamente, (2) se recuperarán datos

del tweet desencadenante y se enviará un correo electrónico personalizado (segunda caja).

Como se puede ver es un flujo de trabajo muy simple y que puede ocupar dos minutos de nuestro tiempo.

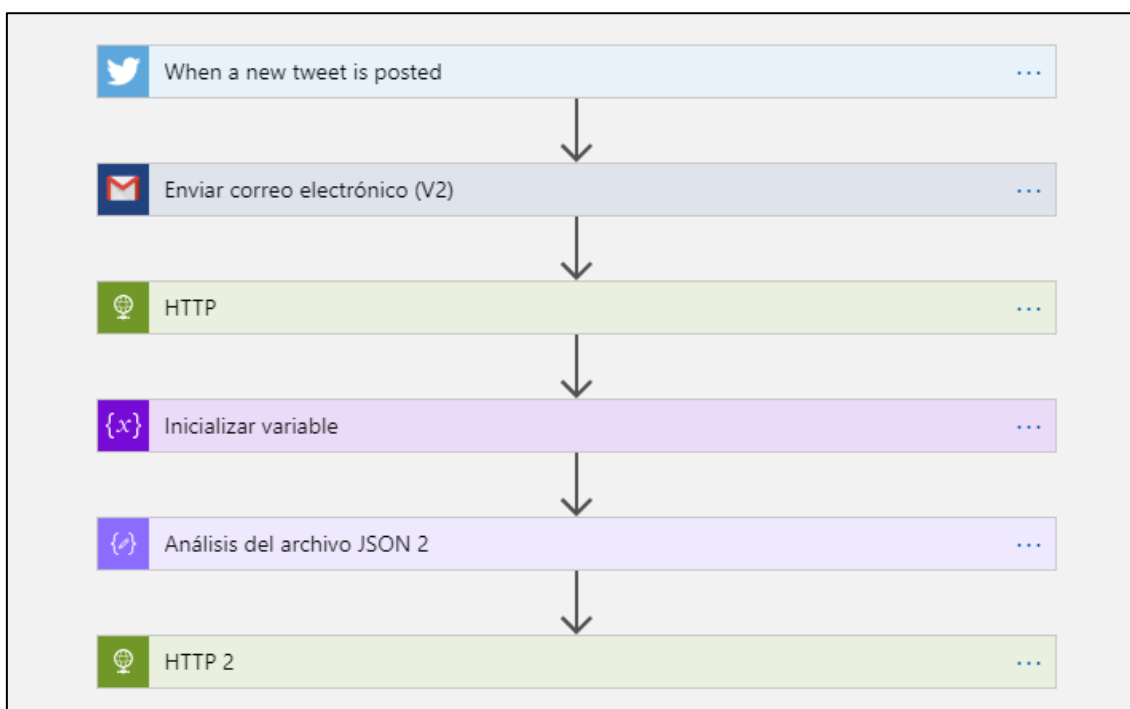


Figura 4. Flujo de trabajo ampliado de ejemplo

En la Figura 4, se aprecia un flujo de trabajo más largo, en el que se añaden 4 cajas. Éstas son responsables de acceder a la API de Google, hacer una búsqueda del contenido del tweet en Google, formatear la respuesta de la API y enviarlo a otra API.

Después de exponer todo lo anterior, uno de los objetivos principales del proyecto ha sido, mediante Logic Apps, recuperar datos de los registros de la plataforma web y ofrecerlos a los usuarios en un formato personalizado, usando plantillas en formatos de texto. Dar a los usuarios la posibilidad de recibir un resumen organizado de sus datos, a modo de suscripción Webhook. [1] [2] [3]

3.4 Monitorización de errores

Por último tema a tratar, como se ha comentado en la introducción, un error en producción, dentro de un ámbito con gran impacto y con muchos usuarios, puede suponer la pérdida de miles de datos y tiempos de retraso irrecuperables. Por eso es necesario un control de errores y una monitorización de estos.

Por ejemplo, en los días de subida a producción, tener un control más exhaustivo de cualquier tipo de error que puede surgir y así poder actuar cuanto antes para solucionar el problema.

4. Metodología

Para el desarrollo del proyecto se ha empleado una metodología de trabajo en cascada pero que ha ido poco a poco modificándose, debido a la corrección de ciertos aspectos durante el desarrollo. En primera instancia, sin embargo, el desarrollo ha sido de manera secuencial, por fases, comenzando por un análisis, diseño y terminando con la puesta en producción y pruebas.

Las ventajas que supone este tipo de metodología son (1) la capacidad de medir de forma más fácil el progreso del proyecto, (2) la planificación es más sencilla, (3) solo hay una persona implicada en el desarrollo y (4) se deben desarrollar varios componentes de software de manera paralela.

Las posibles desventajas de usar esta metodología están relacionadas con las primeras fases, de captación de requisitos y diseño. Es decir, hay que asegurar la correcta realización de las primeras fases para evitar arrastrar el error durante todo el desarrollo. [4] [5]

5. Desarrollo

En este punto se detallan cada una de las fases de desarrollo del proyecto.

5.1 Toma de requisitos

En la toma de requisitos se han clasificado a éstos en tres grupos: funcionales, datos y seguridad.

5.1.1 Requisitos funcionales

Código	RQF - 01
Nombre	Registrar factura
Versión	1.0
Autor	Alex
Tipo	Funcional
Descripción	El usuario debe poder registrar una nueva factura con sus datos pertinentes por parte del usuario.
Prioridad	Alta

Tabla 1: RQF – 01

Código	RQF – 02
Nombre	Registrar envío
Versión	1.0
Autor	Alex
Tipo	Funcional
Descripción	El usuario debe poder registrar un nuevo envío.
Prioridad	Alta

Tabla 2: RQF – 02

Código	RQF – 03
Nombre	Registrar recepción
Versión	1.0
Autor	Alex
Tipo	Funcional
Descripción	El usuario debe poder registrar una nueva recepción.
Prioridad	Alta

Tabla 3: RQF – 03

Código	RQF - 04
Nombre	Registrar pago
Versión	1.0
Autor	Alex
Tipo	Funcional
Descripción	El usuario debe poder registrar un nuevo pago realizado.
Prioridad	Alta

Tabla 4: RQF – 04

Código	RQF - 05
Nombre	Registrar orden
Versión	1.0
Autor	Alex
Tipo	Funcional
Descripción	El usuario debe poder registrar una nueva orden de pedidos.
Prioridad	Alta

Tabla 5: RQF – 05

Código	RQF - 06
Nombre	Registrar EOIDs ¹
Versión	1.0
Autor	Alex
Tipo	Funcional
Descripción	El usuario debe poder crear y registrar nuevos EOIDs.
Prioridad	Alta

Tabla 6: RQF – 06

¹ Identificador utilizado para englobar varias fábricas, máquinas, productos, etc.

Código	RQF - 07
Nombre	Registrar Facilities
Versión	1.0
Autor	Alex
Tipo	Funcional
Descripción	El usuario debe poder registrar una nueva facility.
Prioridad	Alta

Tabla 7: RQF – 07

Código	RQF – 08
Nombre	Registrar Machines
Versión	1.0
Autor	Alex
Tipo	Funcional
Descripción	El usuario debe poder registrar una nueva machine.
Prioridad	Alta

Tabla 8: RQF – 08

Código	RQF - 09
Nombre	Registrar peticiones
Versión	1.0
Autor	Alex
Tipo	Funcional
Descripción	El usuario debe poder registrar nuevas peticiones.
Prioridad	Alta

Tabla 9: RQF – 09

Código	RQF - 10
Nombre	Administrar usuarios
Versión	1.0
Autor	Alex
Tipo	Funcional
Descripción	El administrador del sistema debe poder registrar nuevos usuarios y gestionarlos.
Prioridad	Media

Tabla 10: RQF – 10

Código	RQF - 11
Nombre	Administrar productos
Versión	1.0
Autor	Alex
Tipo	Funcional
Descripción	El usuario debe poder administrar sus productos.
Prioridad	Alta

Tabla 11: RQF – 11

Código	RQF - 12
Nombre	Administrar líneas de producción
Versión	1.0
Autor	Alex
Tipo	Funcional
Descripción	El usuario debe poder registrar una nueva línea de producción.
Prioridad	Media

Tabla 12: RQF – 12

Código	RQF - 13
Nombre	Acceso perfil usuario
Versión	1.0
Autor	Alex
Tipo	Funcional
Descripción	El usuario debe poder acceder a su perfil y editarlo.
Prioridad	Media

Tabla 13: RQF – 13

Código	RQF - 14
Nombre	Ver informes
Versión	1.0
Autor	Alex
Tipo	Funcional
Descripción	El usuario debe poder acceder a los informes que disponen los datos referentes a las facturas, los pagos, los envíos, las recepciones, las entregas las órdenes y los seriales.
Prioridad	Alta

Tabla 14: RQF – 14

Código	RQF - 15
Nombre	Ver informes
Versión	1.0
Autor	Alex
Tipo	Funcional
Descripción	El usuario debe poder disponer los datos en formato PDF, utilizando una plantilla propia, subida al sistema, o a partir de una por defecto.
Prioridad	Alta

Tabla 15: RQF – 15

Código	RQF - 16
Nombre	Monitorizar acciones
Versión	1.0
Autor	Alex
Tipo	Funcional
Descripción	Los desarrolladores deben poder seguir el flujo de acciones realizadas desencadenantes de un error.
Prioridad	Alta

Tabla 16: RQF – 16

5.1.2 Requisitos de datos

Código	RQD - 01
Nombre	Facturas
Versión	1.0
Autor	Alex
Tipo	Datos

Descripción	<p>La información que se dispone de las facturas es la siguiente:</p> <ul style="list-style-type: none"> • Id • EOID • Tipo • Numero • Fecha de factura • Cantidad • Moneda • Id del comprador • Estado • Fecha de creación
Prioridad	Alta

Tabla 17: RQD – 01

Código	RQD - 02
Nombre	Envíos
Versión	1.0
Autor	Alex
Tipo	Datos
Descripción	<p>La información que se dispone de los envíos es la siguiente:</p> <ul style="list-style-type: none"> • Id • Facility Id • Fecha envío • Método de transporte • Vehículo • Estado • Fecha de creación
Prioridad	Alta

Tabla 18: RQD – 02

Código	RQD - 03
Nombre	Recepciones
Versión	1.0

Autor	Alex
Tipo	Datos
Descripción	La información que se dispone de las recepciones es la siguiente: <ul style="list-style-type: none"> • Id • EOID • Facility Id • Fecha de creación
Prioridad	Alta

Tabla 19: RQD – 03

Código	RQD - 04
Nombre	Pagos
Versión	1.0
Autor	Alex
Tipo	Datos
Descripción	La información que se dispone de los pagos es la siguiente: <ul style="list-style-type: none"> • Id • EOID • Fecha de pago • Tipo de pago • Moneda • Cantidad • Pagador Id • Factura Id • Fecha creación
Prioridad	Alta

Tabla 20: RQD – 04

Código	RQD - 05
Nombre	Orden
Versión	1.0
Autor	Alex
Tipo	Datos

Descripción	La información que se dispone de las órdenes es la siguiente: <ul style="list-style-type: none"> • Id • EOID • Fecha orden • Número orden • Fecha de creación
Prioridad	Alta

Tabla 21: RQD – 05

Código	RQD - 06
Nombre	EOID
Versión	1.0
Autor	Alex
Tipo	Datos
Descripción	La información que se dispone de los EOIDS es la siguiente: <ul style="list-style-type: none"> • Id • Nombre • Descripción • Calle • Número de casa • Código postal • Ciudad • País ID • Email • Activo desde • Activo hasta • Habilitado
Prioridad	Alta

Tabla 22: RQD – 06

Código	RQD - 07
Nombre	Facilities
Versión	1.0
Autor	Alex
Tipo	Datos

Descripción	<p>La información que se dispone de las Facilities es la siguiente:</p> <ul style="list-style-type: none"> • Id • EOID • Nombre • Descripción • Calle • Número de casa • Código postal • Ciudad • País ID • Email • Tipo • Activo desde • Activo hasta • Habilitado
Prioridad	Alta

Tabla 23: RQD – 07

Código	RQD - 08
Nombre	Machines
Versión	1.0
Autor	Alex
Tipo	Datos
Descripción	<p>La información que se dispone de las Machines es la siguiente:</p> <ul style="list-style-type: none"> • Id • EOID • Facility ID • Productor • Modelo • Código • Número de serie • Nombre • Descripción • Tipo • Activo desde • Activo hasta • Habilitado

Prioridad	Alta
-----------	------

Tabla 24: RQD – 08

Código	RQD - 09
Nombre	Cientes
Versión	1.0
Autor	Alex
Tipo	Datos
Descripción	<p>La información que se dispone de los clientes es la siguiente:</p> <ul style="list-style-type: none"> • Id • Código • Nombre • Logo • Habilitado • Fecha creación
Prioridad	Alta

Tabla 25: RQD – 09

Código	RQD - 09
Nombre	Usuarios
Versión	1.0
Autor	Alex
Tipo	Datos
Descripción	<p>La información que se dispone de los clientes es la siguiente:</p> <ul style="list-style-type: none"> • Id • Email • Nombre • Estado • Cliente ID • Rol • Contraseña
Prioridad	Alta

Tabla 26: RQD – 10

5.1.3 Requisitos de seguridad

Código	RQS - 01
Nombre	Control acceso
Versión	1.0
Autor	Alex
Tipo	Seguridad
Descripción	<p>El sistema cuenta con dos tipos de accesos:</p> <ul style="list-style-type: none">• Administrador: Tiene acceso a la Web con todas sus funcionalidades, incluidas las de monitorización y creación de usuarios.• Cliente: Tiene acceso a la Web con restricciones.
Prioridad	Alta

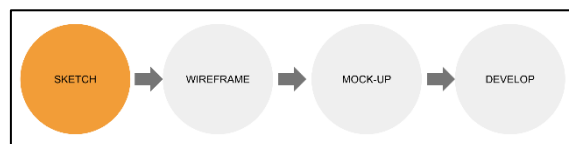
Tabla 27: RQS– 01

5.2 Diseño

El paso que tomar después de la captación de requisitos ha sido el diseño. En todos los proyectos es un paso crucial, ya que antes del desarrollo e implementación de la web se debe tener un mapa global de cómo va a estar organizada ésta, cómo se van a disponer los datos y cómo se va a poder navegar por ella, además de la apariencia. Además, da la posibilidad de hacer cambios en una etapa temprana del proyecto y llegar a una propuesta que satisfaga a los futuros usuarios. [7]

“Los prototipos nos permiten explorar nuestras ideas antes de invertir tiempo y dinero en el desarrollo. Lejos de ser una pérdida de tiempo, trabajar con prototipos nos permiten crear múltiples soluciones para poder *fallar rápido y barato* “²

Para este proceso, se disponen de varias herramientas y métodos que ayudan a desarrollar Wireframes, Mockups y prototipos. [8]



Cabe destacar que el objetivo de la plataforma web ha sido, en todo momento, ofrecer a los usuarios una disposición eficiente de los datos y registros. No se ha buscado que tenga una apariencia destacable y captora de nuevos usuarios y, por lo tanto, se ha obviado el paso del Mockup, que busca precisamente eso.

5.2.1 Wireframe

El Wireframe [9] se ocupa de dar un diseño y definir una estructura básica de la web. Se suele utilizar con escala de grises para no destacar tanto la apariencia y centrar la discusión en el concepto, estructura y componentes básicos. La forma más eficiente y económica de crear un Wireframe es haciéndolo a mano, con papel y lápiz, dando la opción de realizar cambios de forma rápida.

² Casabona, Eugenia (Mar 20, 2019). Prototipos, wireframes, mockups y sketches ¿para qué? Recuperado de <https://medium.com>

Como punto de partida se ha diseñado la interfaz principal con la organización de los ítems de la página y el método de navegación de ésta.

Diseño página principal

En primer lugar, se ha creado la página principal con los elementos *shared* para el resto de las vistas de la web:

- Menú horizontal para seleccionar el idioma, acceder al cierre de sesión y al logo de la empresa, para ir a la página principal.
- Menú vertical navegable para acceder a cada una de las funcionalidades de la web.

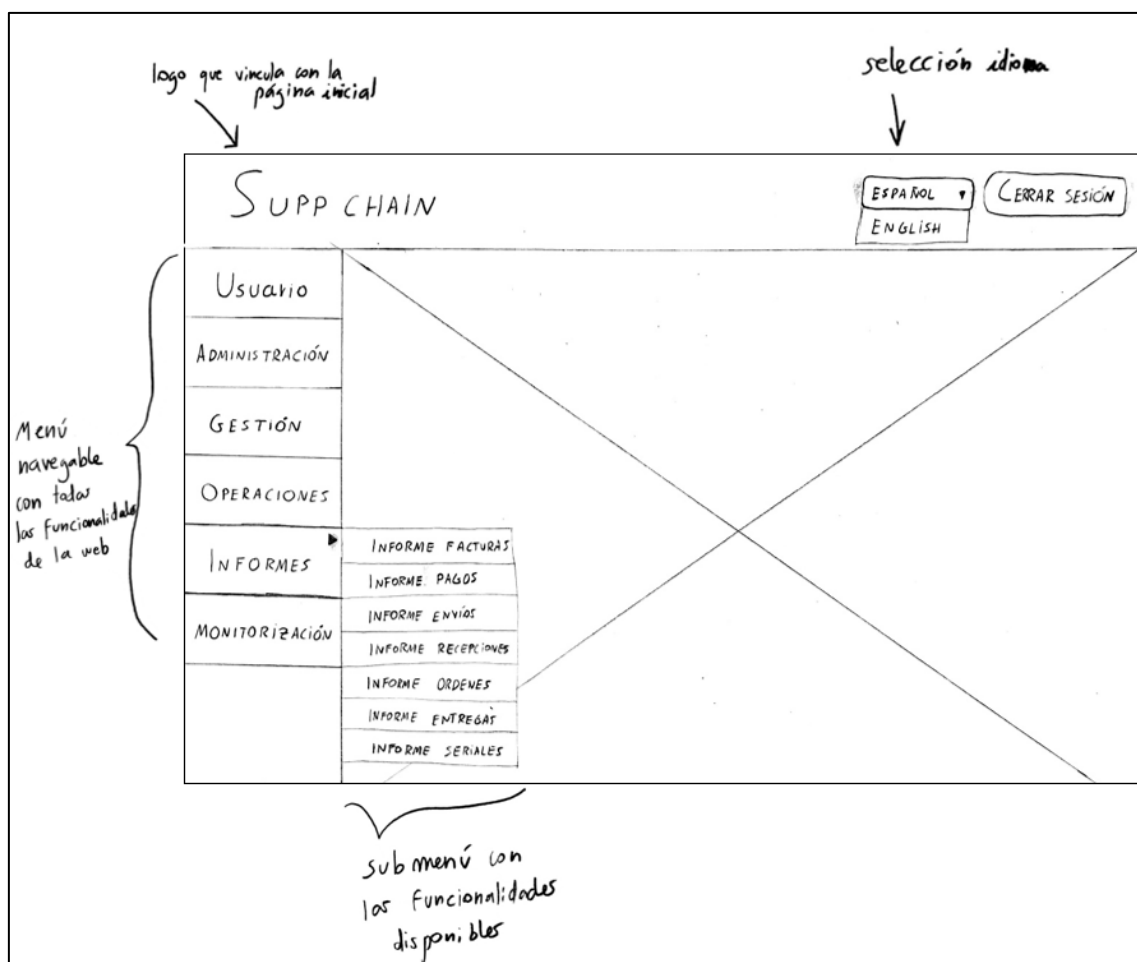


Figura 5. Wireframe página principal

Vista de informe de facturas

En segundo lugar, se ha diseñado una vista de ejemplo de informe de las facturas en el que vemos los elementos *shared*, comentados anteriormente, y el contenido de la vista en cuestión:

- Cajas de texto para aplicar filtros en los resultados que se muestran en la tabla, mostrando los datos relevantes de las facturas.
- Botón adicional en la esquina superior derecha para poder refrescar el contenido.

Filtro fecha desde - hasta

Número	Fecha factura	Tipo	Comproedor	EU	Cantidad
54928125	12/12/2019	Original	Coca Cola	Si	500

Figura 6. Wireframe informe de facturas

5.2.2 Prototipo

El paso siguiente ha sido crear el prototipo de la web. Estos son representaciones de alta fidelidad que simulan la interacción con la interfaz y que permiten al usuario experimentar la experiencia de uso.

Para su creación existen varias herramientas y aplicaciones muy versátiles y con mucha flexibilidad de las que destacan Webflow y Axure. Ésta última, ha sido la escogida en este proyecto, puesto que ofrece a los estudiantes una licencia de forma gratuita. [10]

La aplicación *Axure* es muy completa y ofrece una amplia variedad de opciones, aunque es algo compleja. Permite crear las vistas de la página web una por una u organizadas en carpetas, da la posibilidad de crear plantillas compartidas o elementos *shared*, permite crear todo tipo de interacciones y vínculos y da, en todo momento, la posibilidad de visualizar el resultado en el navegador.

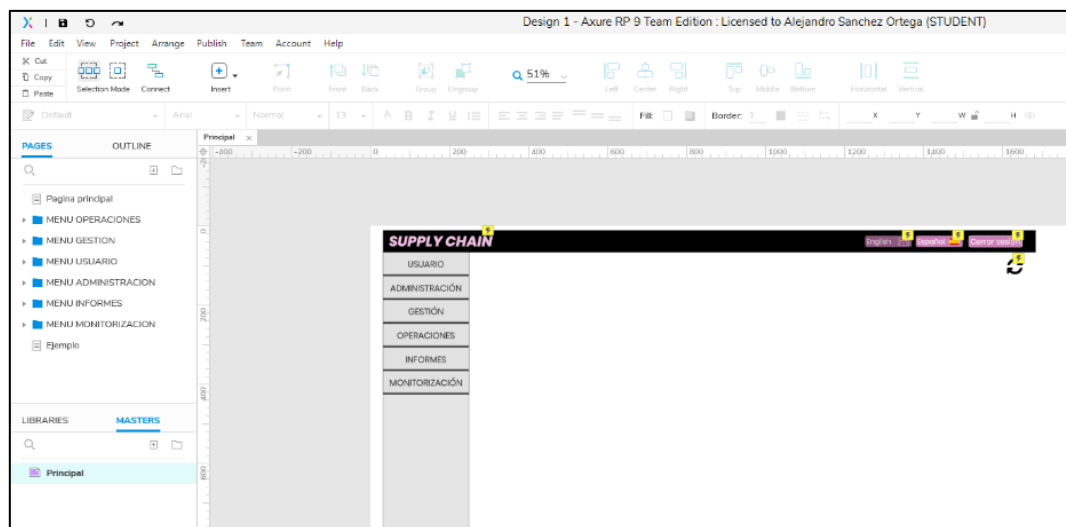


Figura 7. Interfaz de Axure

1. Creación página principal

Como se ha comentado anteriormente, la aplicación da la posibilidad de crear elementos compartidos entre todas las vistas, evitando tener que ir creándolos en cada una de ellas y sin estar sujeto a tener que hacer cambios, si los precisa, en cada una de ellas. Es por eso por lo que lo primordial ha sido crear los menús de la barra superior y la barra lateral.



Figura 8. Página principal

Diseño de las vistas

El paso siguiente, ha sido el diseño de cada una de las vistas que se van a desarrollar en la web. Para ello en el panel lateral de *Axure* se puede organizar tus vistas en carpetas, por ejemplo para organizar las vistas según el ítem del menú seleccionado.

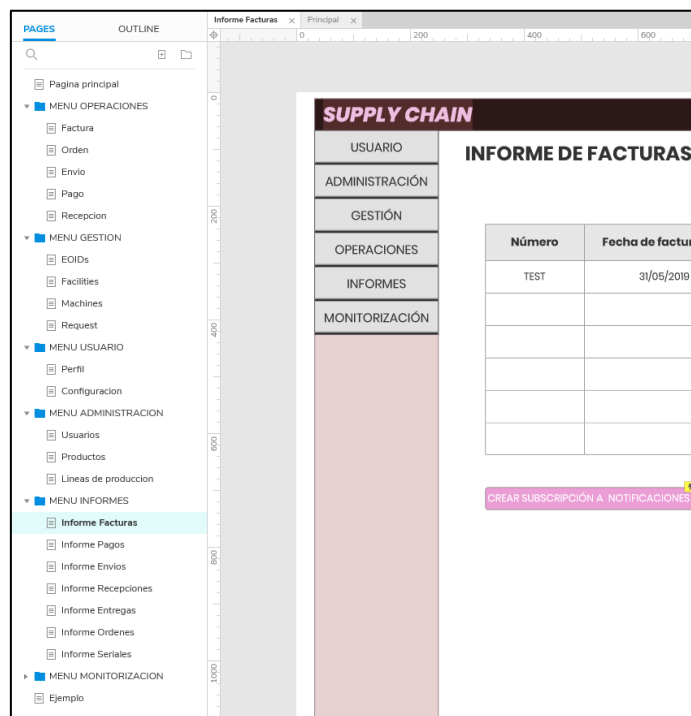


Figura 9. Menú lateral Axure

Una vez creadas todas las vistas, se han asignado las interacciones entre los elementos de las vistas y los vínculos entre los diferentes ítems del menú. Por ejemplo, estas han sido las interacciones entre escoger el idioma en Inglés o Español.



Figura 10. Resultado de la traducción en la web

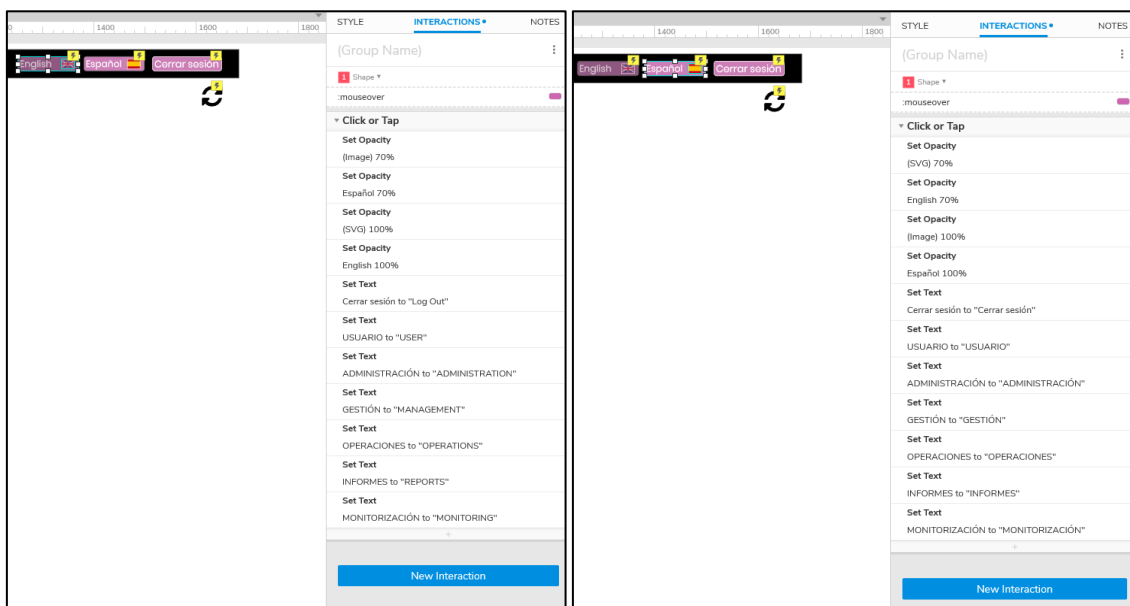


Figura 11. Interacciones de los botones Español-English

Después del diseño completo y la asignación de interacciones y vinculaciones, se visualiza el resultado en el navegador.

SUPPLY CHAIN

EnglishEspañolCerrar sesión

USUARIOADMINISTRACIÓNGESTIÓNOPERACIONESINFORMESMONITORIZACIÓN

REGISTRAR FACTURA

NÚMERO285396174FECHA DE FACTURACIÓNdd/mm/aaaa

PRECIO NETO2275€MONEDAEUR

COMPRADOR

☒EUROPA

COMPRADOR IDM12AP7C58

PRODUCTOS

SERIALES

AÑADIR

BORRAR

CANCELAR

REGISTRAR

SUPPLY CHAIN

EnglishEspañolCerrar sesión

USUARIOADMINISTRACIÓNGESTIÓNOPERACIONESINFORMESMONITORIZACIÓN

PERFIL

NOMBRETEST

E-MAILtest@test.com

STATUSHABILITADO

IDIOMAESPAÑOL

ROLADMINISTRADOR

SUPPLY CHAIN

EnglishEspañolCerrar sesión

USUARIOADMINISTRACIÓNGESTIÓNOPERACIONESINFORMESMONITORIZACIÓN

NEW REQUEST

EIDsTEST-12345FIDsFRMachinesColaCooler

Producto

PRODUCTOS EXISTENTESCOLA

TIPOREFresco

DESCRIPCIÓNREFresco DE COLA

MERCADOFranciaRUTAEspaña

CANTIDAD20MANUFACTURING LINEPL1

COMENTARIOSCOMENTARIOS...

CANCELAR

ENVIAR REQUEST

Página 32 | 85

SUPPLY CHAIN
English 
Español 
Cerrar sesión

USUARIO
ADMINISTRACIÓN
GESTIÓN
OPERACIONES
INFORMES
MONITORIZACIÓN

INFORME DE FACTURAS



Número	Fecha de facturación	Tipo	Comprador	EU	Cantidad	...
TEST	31/05/2019	Original	CocaCola	Si	500	

CREAR SUBSCRIPCIÓN A NOTIFICACIONES

Figuras 12: Vistas del diseño

Por otro lado, Axure da la posibilidad de compartir un proyecto o diseño con otros usuarios y facilita el trabajo paralelo entre miembros de un mismo equipo.

El enlace del diseño es <https://qeijfn.axshare.com> (contraseña: TFGALEX) .

5.3 Creación de la plataforma web

Para implantar la información obtenida de los requisitos y orquestada en el diseño, en esta fase se detalla los pasos seguidos durante la programación de la plataforma web.

Como punto de partida, se ha establecido DDD [12] como enfoque de desarrollo del sistema en una arquitectura de web MVC.

Las ventajas de utilizar este enfoque es que se obtiene una mejor perspectiva a nivel de colaboración entre expertos del dominio y los desarrolladores, para concebir un software con los objetivos bien claros.

La estructura de DDD se aprecia a continuación:

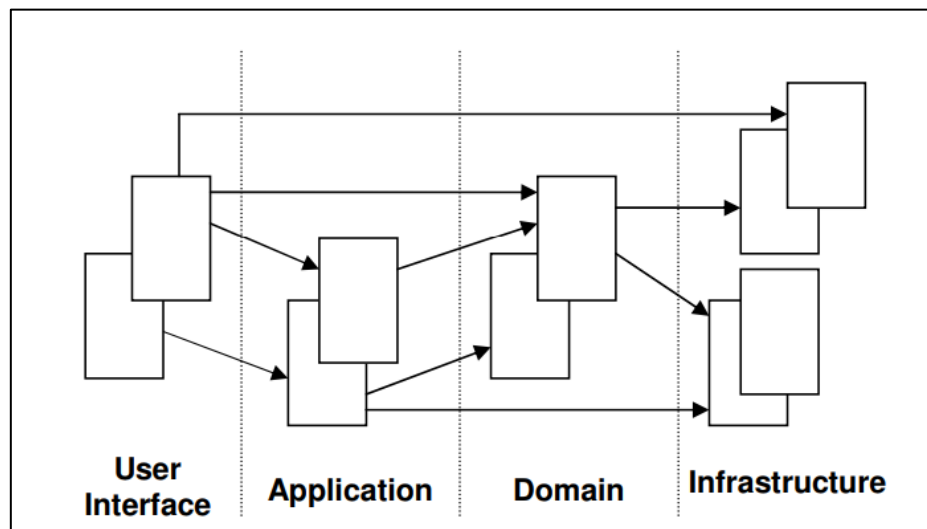


Figura: estructura DDD

El concepto principal de DDD es que el Dominio quede representado de forma clara y concisa en la estructura del software. Los modelos y reglas de negocio se incluyen en el core de la aplicación.

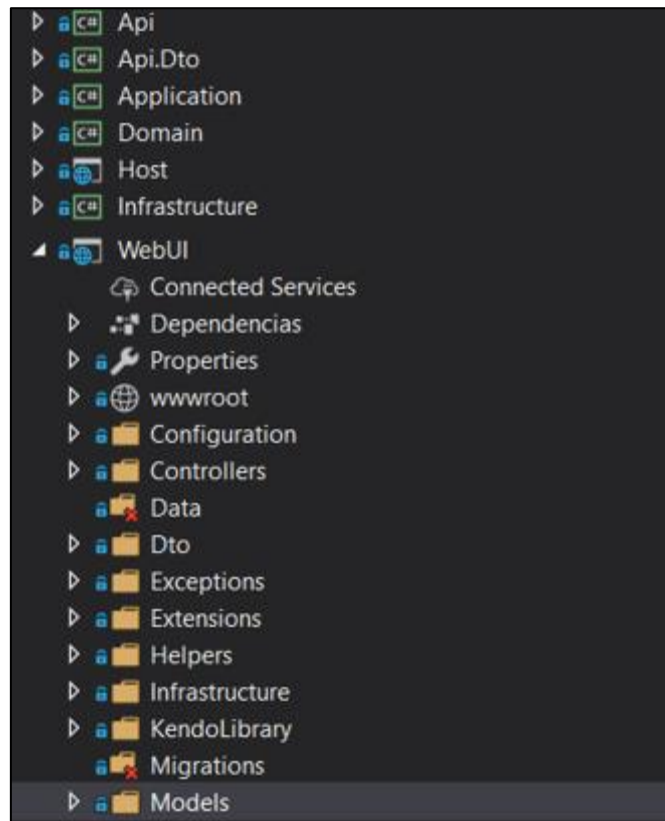


Figura: Estructura del código

Como se aprecia en la Figura 14, la estructura del código se divide en cuatro puntos, a parte del host y la API.

- Application se encarga de realizar las operaciones entra el Domain y la Web
- Domain representa todas las entidades necesarias para representar los modelos y reglas de negocio.
- Infrastructure se encarga de acceder a bases de datos y elementos externos.
- WebUI es la interfaz con la que el usuario interactúa y se encarga de mostrar todos los datos.

5.4 Configuración de la base de datos

Como se ha comentado anteriormente, Microsoft Azure da la posibilidad de acceder a numerables recursos y servicios. El servicio empleado para la creación de la base de datos ha sido SQL Database y junto con SQL Server.

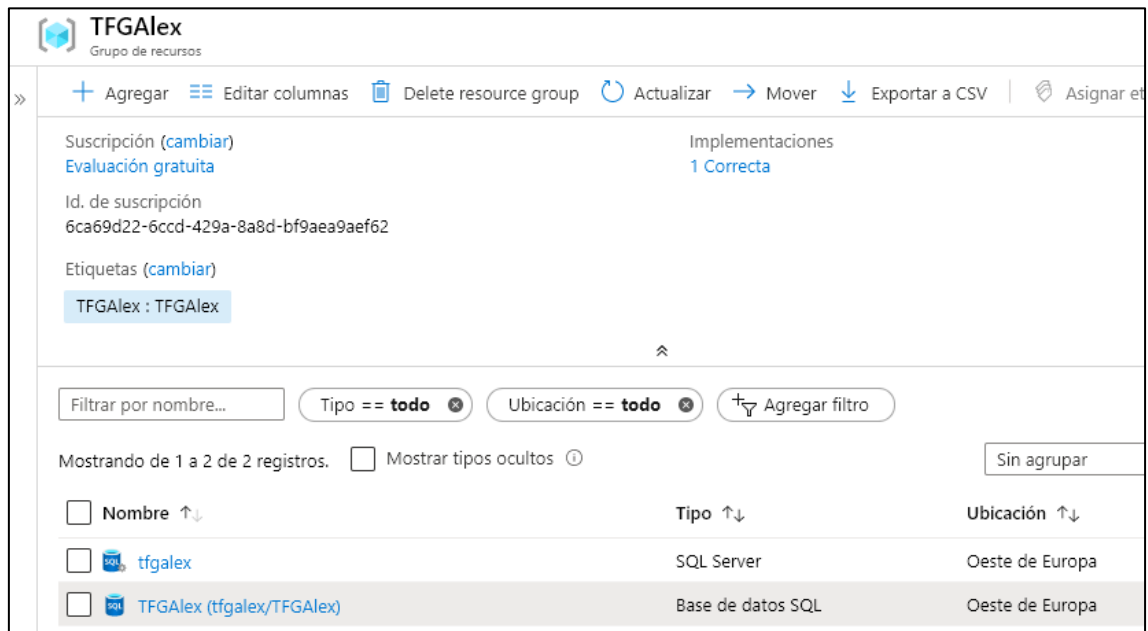


Figura 13. Creación recurso BD en Azure.

El DTU³ y el almacenamiento es de 10 y 250GB respectivamente. Se ha escogido un tipo de servicio estándar puesto que el proyecto no manejará grandes cantidades de datos como, por otro lado, sí lo haría en la práctica.

En la Figura 13 se observa la vista de Microsoft Azure con el grupo de recursos⁴:

- tfgalex → SQL Server
- TFGAlex → Base de datos SQL

³ ¿Qué es DTU? Recuperado de <https://docs.microsoft.com/es-es/azure/sql-database/sql-database-service-tiers-dtu>

⁴ Un grupo de recursos es un contenedor que almacena los recursos relacionados con una solución de Azure. Recuperado de <https://docs.microsoft.com/es-es/azure/azure-resource-manager/management/manage-resource-groups-portal>

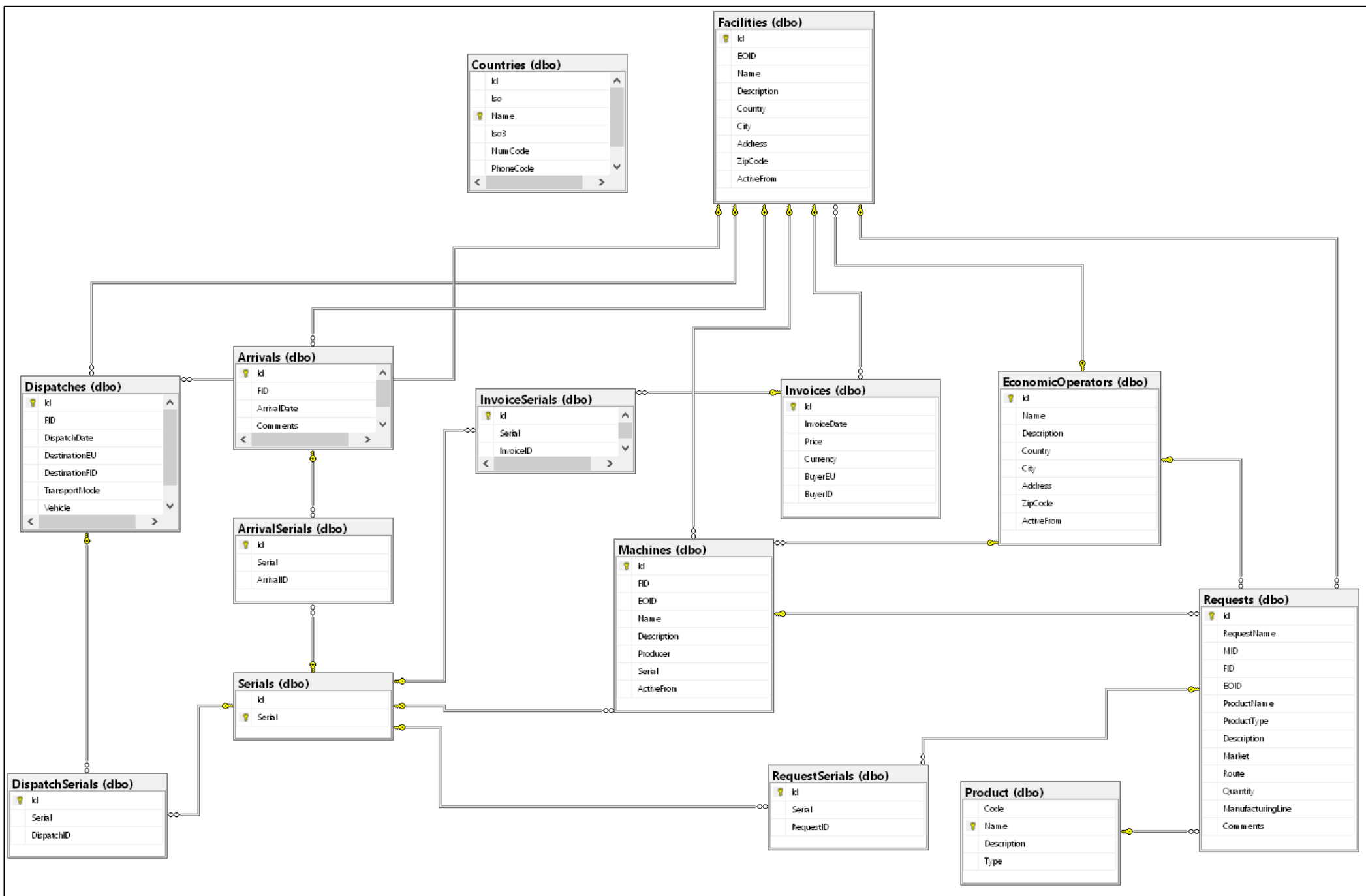


Figura 15. Diagrama de la base de datos.

5.5 Configuración de la API

Para la realización de cualquiera de las acciones de nuestra web se ha de interactuar con la base de datos para leer, editar o eliminar cualquier tipo de registro. De la misma manera y como previamente se ha comentado, Logic Apps tiene una gran flexibilidad y eficiencia a la hora de trabajar con llamadas HTTP y es por eso por lo que se necesita la creación de una API.

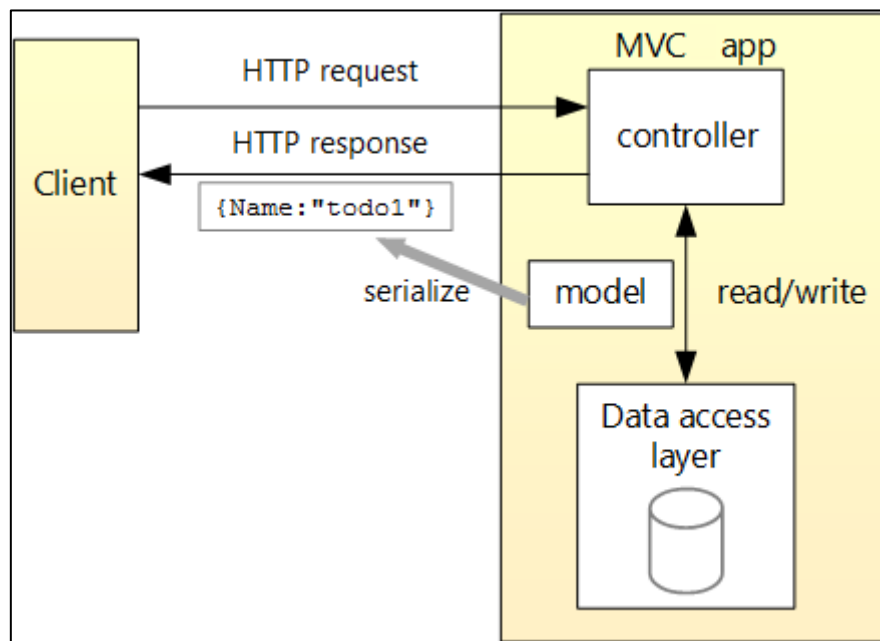


Figura 16. Esquema de funcionamiento.

Una API es un conjunto de definiciones y protocolos que se emplean para desarrollar e integrar el software de las aplicaciones. Permiten que los productos y servicios se comuniquen con otros sin tener la necesidad de saber cómo están implementados. A parte de la simplicidad que esto aporta, supone un ahorro de tiempo y dinero.

5.5.1 API web con ASP.NET

Para satisfacer las solicitudes de obtener, editar y eliminar datos de la SQL Database, sobre SQL Server, se ha implementado la API web sobre ASP.NET Core para que sea compatible con el front-end, es decir la plataforma web.

Al haber desarrollado bajo el enfoque de DDD y poder respetar la independencia de cada una de las partes, la API REST se encarga únicamente de recibir peticiones del front-end y ejecutar, mediante el patrón de diseño *Mediator*, llamadas a la capa de Application, que posteriormente accederá a Infrastructure para acceder a la base de datos.

Más adelante se concretará con ejemplos el flujo de ejecución del back-end.

5.5.2 Puntos de acceso API

Para satisfacer todas las solicitudes del front-end para obtener elementos de la base de datos o actuar en ella (crear, actualizar o eliminar), existen una serie de puntos de acceso organizados según las tablas en las que se requiere acceso.

Como vemos en la imagen a continuación, los end-points se dividen en cinco archivos que actúan de controladores:

- **Arrival**
 - Para registrar o consultar las recepciones.
- **Dispatch**
 - *Para registrar o consultar los envíos.*
- **Facility**
 - *Para registrar o consultar las Fábricas o instalaciones*
- **Invoice**
 - *Para registrar o consultar las facturas*
- **Serial**
 - *Para consultar los seriales, asociados a productos.*

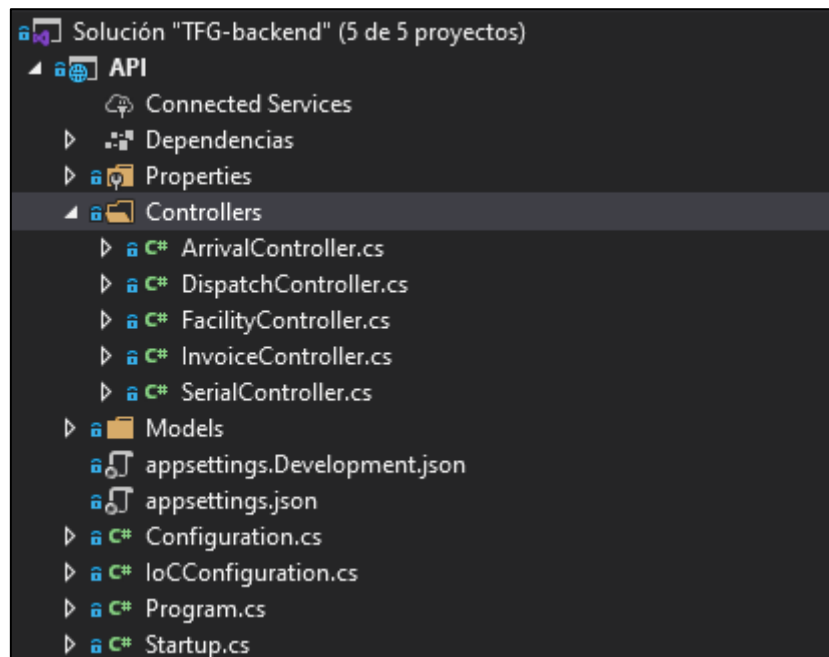


Figura 17. Organización end-points.

Arrival

Para entender el funcionamiento de éste y poder ver como se aplica el DDD en todo el proyecto, más adelante se puede observar un esquema del flujo de trabajo.

En la *figura 21*, se observa la inicialización de la clase *ArrivalController*, la definición de un punto de acceso llamado *Summary* y la aplicación del patrón Mediator:

En las líneas 16 y 17, se observa en primer lugar, el establecimiento de la ruta de acceso y un Decorator que indica que el archivo es un *ApiController* y todo lo que derive en él servirá como *HTTP Api responses*.

De las líneas 20 a las 25, se introducen dos patrones de diseño combinados:

- **Inyección de dependencia**
 - Es encargado de suministrar objetos a una clase en lugar de ser la propia clase que cree dichos objetos. Este patrón es parte de uno

de los cinco principios de diseño de clases conocido como SOLID [14][15]. La *figura 18* muestra un ejemplo del flujo de acción.

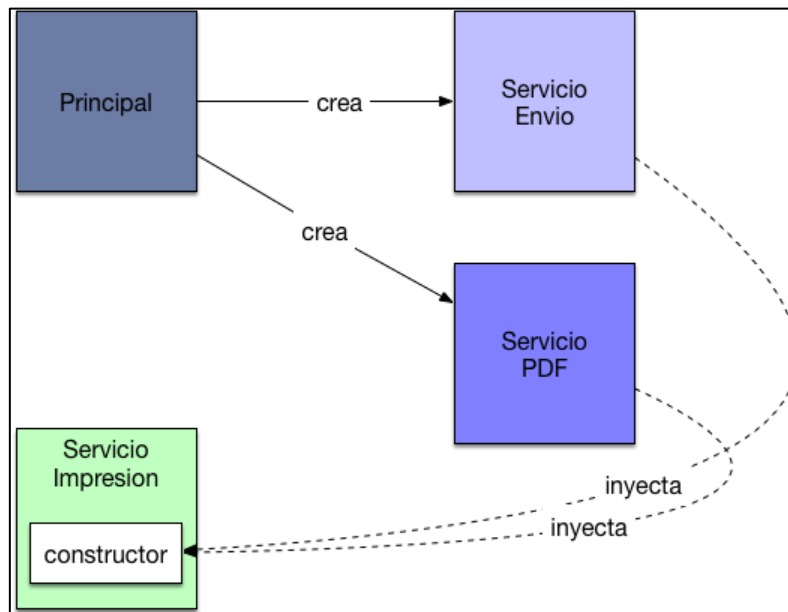


Figura 18. Ejemplo inyección de dependencia.

- En este ejemplo se muestra cómo, desde una clase principal, podemos utilizar dos métodos implementados en la clase Servicio Impresión, sin tener que instanciar el objeto servicio de impresión.
- **Mediator[16]**
 - Es el encargado de comunicar a los Handlers la operación a realizar, pasándoles la información necesaria para realizarla.
 - Como se ha explicado anteriormente, este patrón se utiliza mediante la inyección de dependencia.

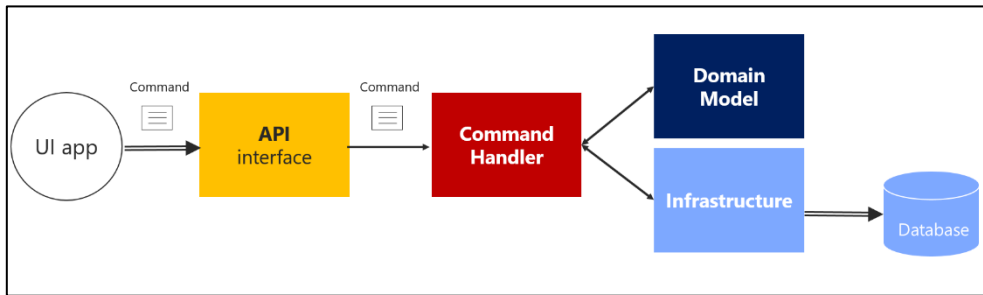


Figura 19. Esquema Mediator DDD

- Mediante un Command Handler (el receptor del command generado en el controlador), la API es capaz de pasar la información necesaria para obtener los datos o actuar sobre las tablas concretas de la Base de datos, respetando la independencia entre cada uno. La Figura 19 muestra el flujo de trabajo del patrón *Mediator*.

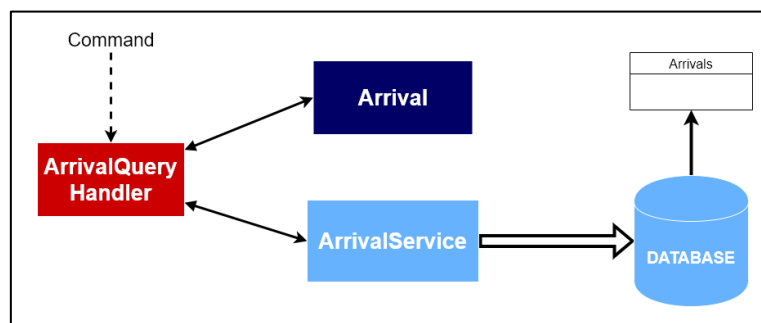


Figura 20. Esquema Mediator DDD Arrival Summary[18]

- En el caso del controlador Arrival, como vemos en la figura 20 y siguiendo el código de la línea 32 de la figura 21, el método Summary recibe una *query* con los parámetros que necesita el usuario para acceder a la base de datos. En el método se formatea las variables y se crea el objeto que se necesita para pasar la información (línea 39) . A continuación, (línea 46) se utiliza el mediator inicializado con la inyección de dependencia para enviar la *ArrivalQuery* a su Handler, llamado *ArrivalQueryHandler*.

El Handler del ArrivalQuery accederá a Infrastructure y a sus servicios implementados y así poder acceder a la base de datos.

```

16 [Route("api/arrival")]
17 [ApiController]
18 public class ArrivalController : ControllerBase
19 {
20     private readonly IMediator _mediator;
21
22     public ArrivalController(IMediator mediator)
23     {
24         _mediator = mediator;
25     }
26
27     [HttpGet]
28     [Consumes("application/json")]
29     [Produces("application/json")]
30     [Route("Summary")]
31     [ProducesResponseType(typeof(PaginatedList<ArrivalResult>), 200)]
32     public async Task<IActionResult> Summary([FromQuery] ArrivalQueryString queryString)
33     {
34         if (!ModelState.IsValid)
35         {
36             return BadRequest(ModelState);
37         }
38
39         var query = new ArrivalQuery(
40             queryString.CreationDateFrom,
41             queryString.CreationDateTo,
42             queryString.Id.Value,
43             queryString.FID
44         );
45
46         var result = await _mediator.Send(query);
47         return Ok(result);
48     }

```

Inyección de dependencia del patrón Mediator

Creación de la query

Uso del Mediator para enviar la Query a ArrivalQueryHandler

Figura 21. Arrival API

```

1 referencia
public class ArrivalQueryHandler : IRequestHandler<ArrivalQuery, PaginatedList<ArrivalResult>>
{
    private readonly IArrivalService _arrivalService;
    public ArrivalQueryHandler(IArrivalService arrivalService)
    {
        _arrivalService = arrivalService;
    }
    6 referencias
    public async Task<PaginatedList<ArrivalResult>> Handle(ArrivalQuery request, CancellationToken cancellationToken)
    {
        var filter = GetFilter(request);
        var arrivals = await _arrivalService.GetArrivals(filter);
        int total = await _arrivalService.GetTotalArrivals(filter);
        PaginatedList<ArrivalResult> result = CreateResultResponse(request, arrivals);
        result.Total = total;
        return result;
    }

```

Interfaz de infrastructure para Arrival

Inyección de dependencia ArrivalService

Handler que recibe el send, proveniente de la api

Acceso a Infrastructure y a la base de datos

Figura 22. Handler de Arrival.

5.5.3 Listado puntos de acceso

Como se ha explicado previamente con el caso del Arrival Summary, todos los puntos de acceso de la API, o métodos, siguen el mismo flujo de trabajo. El conjunto de ellos se recoge en el siguiente listado, en el que Summary es un GET que devuelve un resumen y Create, un POST para registrar nuevos:

- Arrival
 - Summary
 - Create
- Dispatch
 - Summary
 - Create
- Facility
 - FIDS
 - Devuelve un listado de todos los FIDS disponibles.
- Invoice
 - Summary
 - Create
- Serial
 - Summary

5.5.4 Pruebas y resultados

Para ayudar en el desarrollo y comprobar que la API funciona correctamente, es decir recibe el formato de petición correctamente y devuelve el resultado esperado, se ha utilizado Postman[17], una extensión del navegador Google Chrome, que permite el envío de peticiones HTTP REST sin necesidad de ejecutarlas desde un cliente.

Una vez se ejecuta la API, ésta permanece a la espera de recibir algún tipo de llamada HTTP, como se ve a continuación:

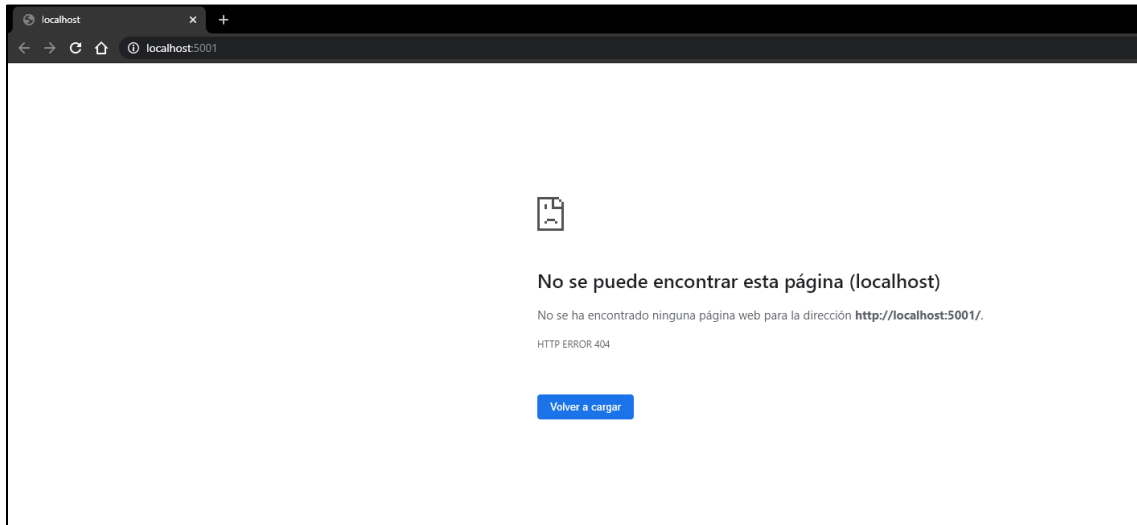


Figura 23. Api en ejecución

A continuación, se puede observar dos ejemplos de llamadas HTTP, un GET, para el summary de Arrivals, y un POST, para el registro de Arrival.

GET SummaryArrival

En la siguiente *figura* 24, se muestra la ejecución de la llamada a ***api/Arrival/Summary*** con sus pertinentes parámetros:



Figura 24. Uri y parámetros

El siguiente paso es ejecutar la llamada y recibir una respuesta:



Figura 25. Respuesta API

Como se observa, en el body con la respuesta aparecen los Arrivals registrados hasta el momento (en este caso 1). El tiempo de respuesta ha sido 1956 ms y el código 200 OK.

POST RegisterArrival

Para el caso de registrar un nuevo Arrival, se genera un body con la información necesaria:

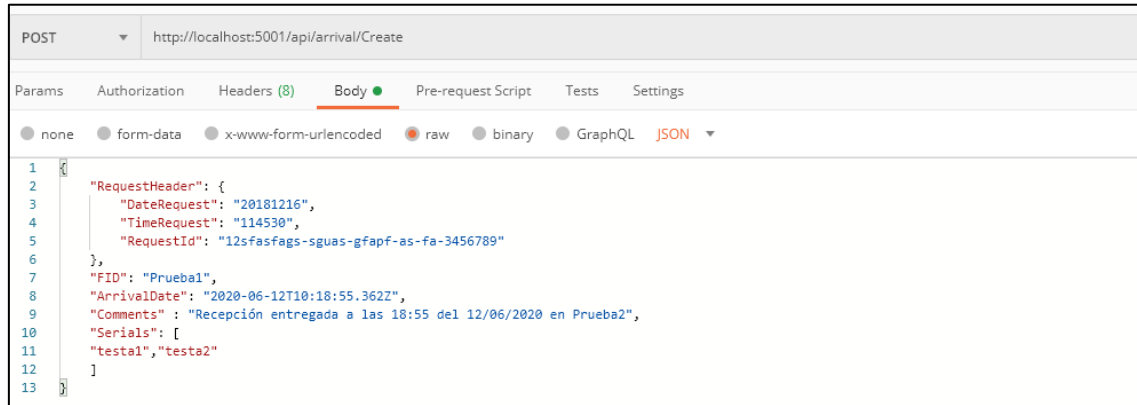


Figura 26. URI, Body POST

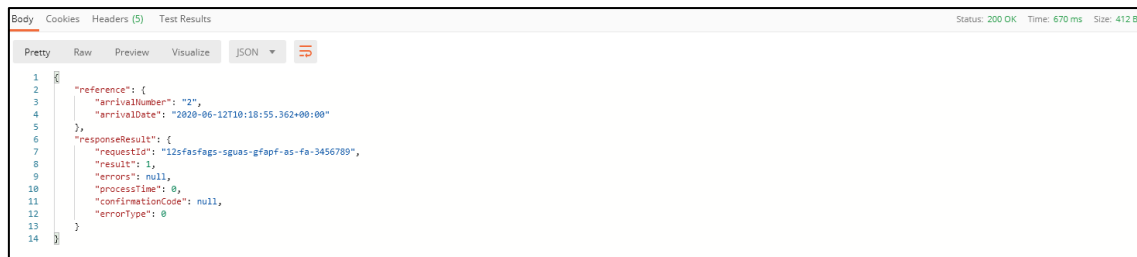
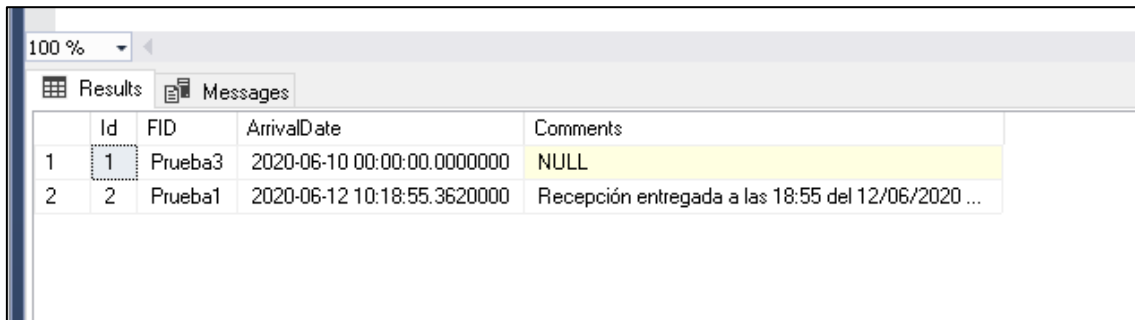


Figura 27. API Response

Finalmente, como se observa, la operación ha tardado 670 ms y el contenido de la respuesta indica si ha habido errores o no y ocupa 412 B.

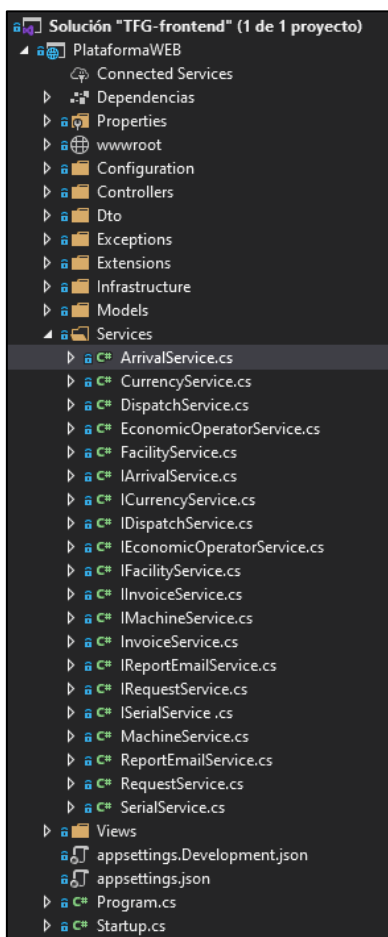
Comprobando la base de datos, se confirma que el registro se ha realizado correctamente:



	Id	FID	ArrivalDate	Comments
1	1	Prueba3	2020-06-10 00:00:00.0000000	NULL
2	2	Prueba1	2020-06-12 10:18:55.3620000	Recepción entregada a las 18:55 del 12/06/2020 ...

Figura 28. Base de datos Arrivals.

5.5.5 Llamada a la API desde Front-end



Una vez finalizado el desarrollo de la API, sus puntos de acceso y sus end-points, el paso siguiente ha sido la implementación de las llamadas desde el front-end.

Como se observa en la *figura 29*, la carpeta Service contiene todos los servicios que engloban las llamadas a la API. Estos servicios serán llamados desde el controlador, que posteriormente le pasará los datos a la vista concreta.

Figura 29. Distribución código Front-end

Siguiendo el ejemplo mencionado anteriormente, en la *figura 30* se observa como el service de Arrival llama a la Api para obtener un resumen de todos los Arrival registrados en la base de datos.

```
public class ArrivalService : IArrivalService
{
    private HttpClient _httpClient;
    private readonly string _remoteServiceBaseUrl;
    private static string _connectionOptions;

    0 referencias
    public ArrivalService(HttpClient httpClient, IOptions<AppSettings> settings, ConnectionOptions connect
    {
        _httpClient = httpClient;
        _remoteServiceBaseUrl = connectionOptions.apiDevelop + "/api/arrival";
    }

    2 referencias
    async public Task<ArrivalResponse> Register(Arrival arrival)...

    1 referencia
    public async Task<PaginatedList<ArrivalReport>> GetArrivals()...

    3 referencias
    public async Task<PaginatedList<ArrivalReport>> GetFilteredArrivals(ArrivalFilters filters)
    {
        var uri = API.Arrival.GetFilteredArrivals( remoteServiceBaseUrl, filters);
        var response = await _httpClient.GetAsync(uri);
        if (response.StatusCode == System.Net.HttpStatusCode.InternalServerError)
        {
            throw new Exception("Error obteniendo las recepciones");
        }
        response.EnsureSuccessStatusCode();
        var jsonResult = await response.Content.ReadAsStringAsync();
        var result = JsonConvert.DeserializeObject<PaginatedList<ArrivalReport>>(jsonResult);
        return result;
    }
}
```

genera uri del punto de acceso de la API

llamada API

comprueba que la respuesta es correcta (HTTP 200)

Figura 30. Servicio para obtener los Arrivals.

Una vez se realiza la llamada y la API genera un resultado, se procesa éste.

```
public static IServiceCollection AddHttpClientServices(this IServiceCollection services, IConfiguration configuration)
{
    services.AddHttpClient<IEconomicOperatorService, EconomicOperatorService>();
    services.AddHttpClient<IFacilityService, FacilityService>();
    services.AddHttpClient<IMachineService, MachineService>();
    services.AddHttpClient<IRequestService, RequestService>();
    services.AddHttpClient<IArrivalService, ArrivalService>();
    services.AddHttpClient<IInvoiceService, InvoiceService>();
    services.AddHttpClient<IDispatchService, DispatchService>();
    services.AddHttpClient<ISerialService, SerialService>();
    services.AddHttpClient<IReportEmailService, ReportEmailService>();

    return services;
}
```

Figura 31. Conjunto de endpoints del Front-end

En la figura 31 se observan todos los end-points del frontend. Concretamente hay servicios para cada una de las entidades.

5.6 Deploy

Como se ha explicado en los puntos anteriores y una vez implementado el front y el back end, el siguiente paso ha sido desplegar la plataforma y la API para darles una disponibilidad global de forma externa. [20]

Como otros componentes del proyecto, se ha utilizado un servicio de Microsoft Azure, en este caso App Service. Con este servicio, se puede realizar una migración y traer el código del proyecto a la nube, sin aplicar cambios en él.

Entre las características más destacables de este servicio se encuentran la fácil gestión y administración, mediante la interfaz de recursos de Azure, y la integración y despliegue continuo CI/CD, que se comentará más adelante.



Figura 32. App Services

Como se observa en la *figura 32*, se han creado dos App Services para mantener la independencia de ambos componentes, uno para el frontend y otro para el backend.

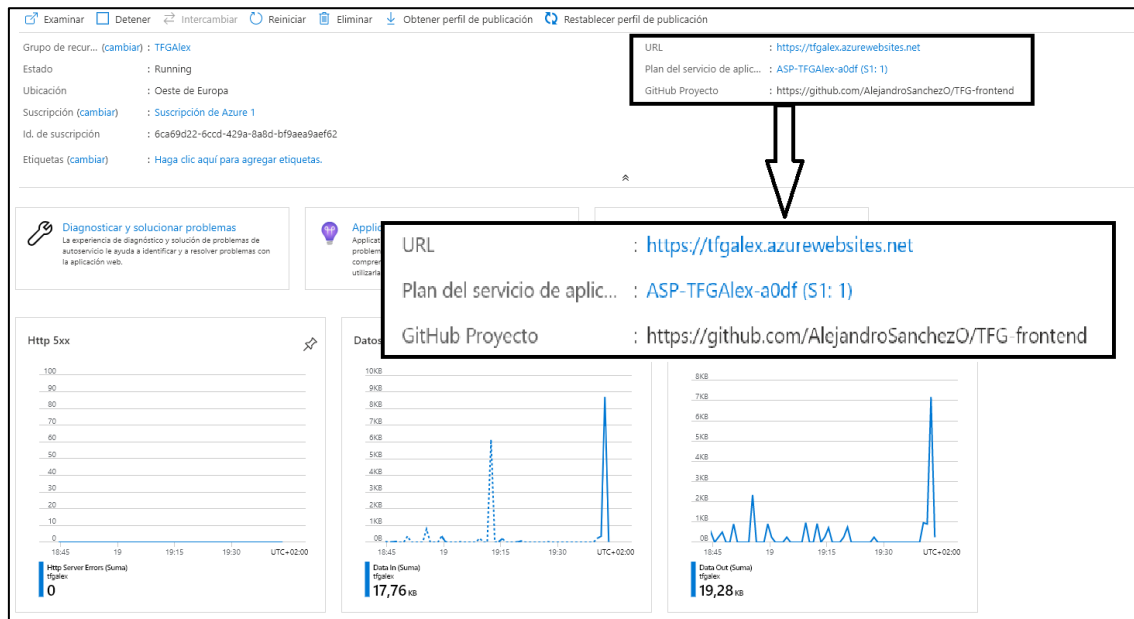


Figura 33. Información App Service frontend

En la figura anterior se observa la información general una vez el App Service está creado. Como se observa, la URL donde se ha desplegado <https://tfgalex.azurewebsites.net>.

Para el despliegue continuo, App Service da la opción de asignarse a un repositorio. Cualquier push que se efectúe en <https://github.com/AlejandroSanchezO/TFG-frontend> hará que el App Service implemente el proyecto automáticamente. Para ello, en el repositorio hay dos ramas, una llamada *master* donde se desarrolla y otra de *deploy*, que está conectada con el App Service.

Desde la interfaz de App Service se pueden gestionar las versiones implementadas del proyecto. La siguiente figura muestra los últimos cambios efectuados:

Origen GitHub	Repositorio https://github.com/AlejandroSanchezO/TFG-frontend		
Compilación Kudu	Rama Deploy	Tipo de control de código fuente Git	
TIEMPO	ESTADO	IDENTIFICADOR DE CONFIRMACIÓN (AU	MENSAJE DE INSERCIÓN EN EL REPOSITORIO
Thursday, June 25, 2020			
8:49:21 PM GMT+2	Correcto (Active)	bd89590 (AlejandroSanchezO)	Invoice filter doc suscription
10:51:26 AM GMT+2	Correcto	885bd5d (AlejandroSanchezO)	no message
6:47:55 AM GMT+2	Correcto	4f9d51a (AlejandroSanchezO)	no message

Figura 34. Implementaciones App Service.

El github con el frontend es <https://github.com/AlejandroSanchezO/TFG-frontend>.

El backend se encuentra en <https://github.com/AlejandroSanchezO/TFG-backend>.

5.7 Logic Apps

Como se ha explicado al inicio de este documento, la segunda parte del proyecto ha consistido en automatizar procesos de generación de informes de los registros a partir del uso de Logic Apps. [19]

5.7.1 Funcionamiento y flujo

Para entender el flujo de operación de las Logic Apps generadas, se comentará un caso específico, ya que todas funcionan de forma similar pero acceden a entidades diferentes.

LogicApps para las facturas

El objetivo es, una vez aplicados los filtros en la pantalla de informes de facturas, poder generar archivos *docx* que recojan los resultados obtenidos del filtro. El documento generado se enviará por correo electrónico a la dirección que se facilite. Además, como modo de suscripción, se puede especificar la periodicidad con la que se quiere recibir este documento, con los resultados actualizados. Es decir, se puede generar la suscripción a un tipo de informe para recibir las facturas filtradas en un intervalo de tiempo a seleccionar.

Para ello, se emplean tres tipos de Logic Apps conectadas entre ellas.

UpdateDocInvoices

Esta Logic App es la encargada de ser el punto de conexión con la plataforma web.

Como vemos en la figura siguiente, la acción desencadenante, es decir que ejecuta el flujo, es una llamada POST, que recibe la solicitud.

Seguidamente, se genera el JSON con la petición. Para generar el archivo docx, se tiene que rellenar una plantilla que, con Visual Basic[21]⁵, completa los campos con el JSON del POST. Una vez generado el documento nuevo se guarda en Google Drive.

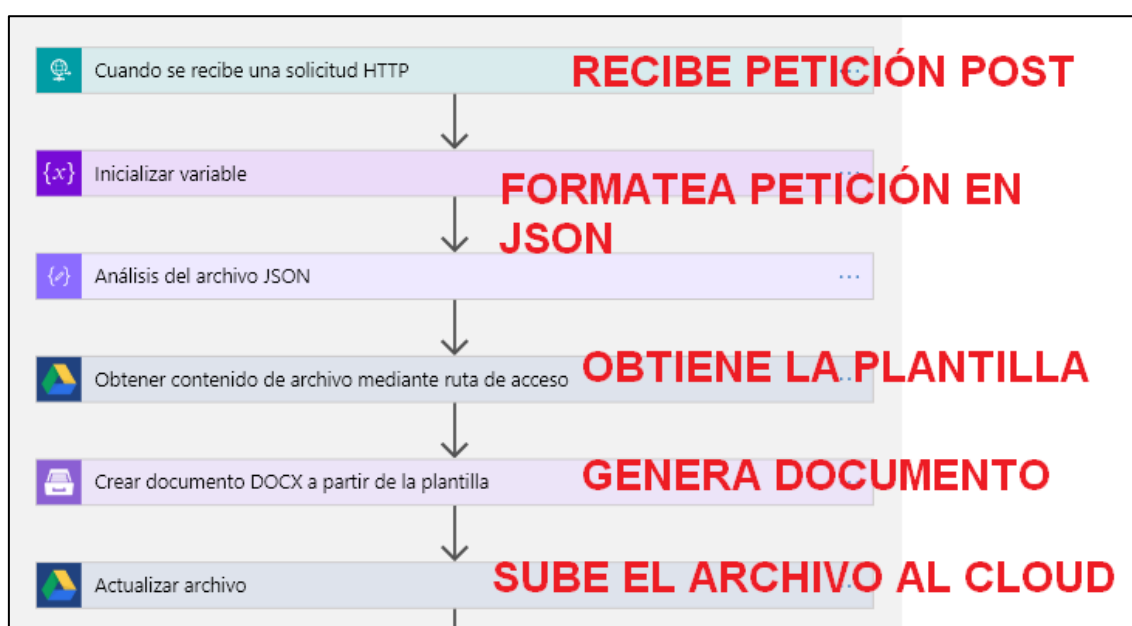


Figura 35. UpdateDocInvoices ½

Después de generar el documento, se envía éste por correo electrónico a la dirección especificada.

La segunda parte de este flujo es el encargado de comprobar si se ha especificado una suscripción al informe.

En ese caso entra en un bucle y llamada a la segunda Logic App, explicada a continuación.

⁵ Se ha empleado la API de *Plumsail*, disponible en la bibliografía.

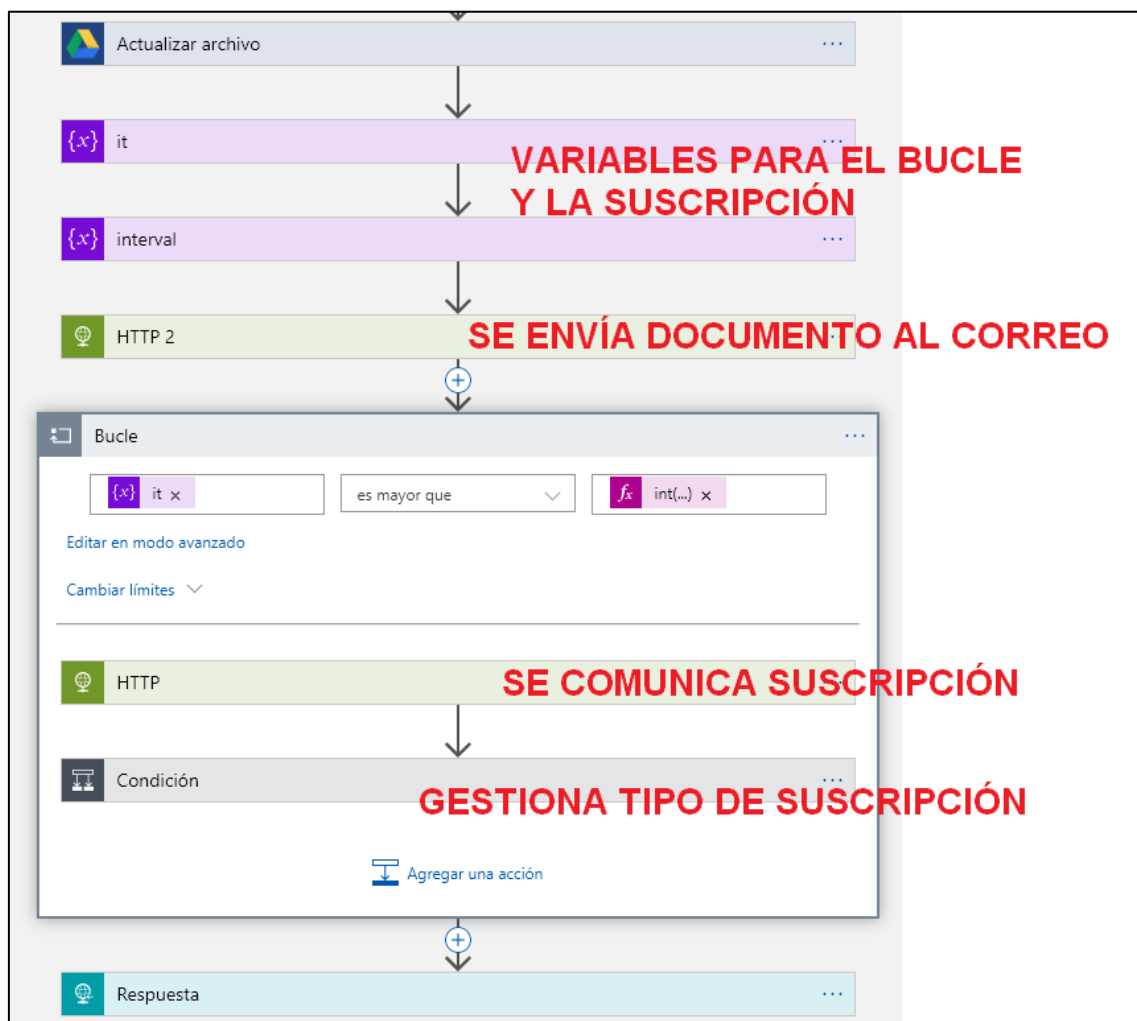


Figura 36. UpdateDocInvoices 2/2

BucleInvoiceSummary

Esta segunda Logic App se activa cuando se especifica la periodicidad que se desea recibir el informe al correo electrónico.

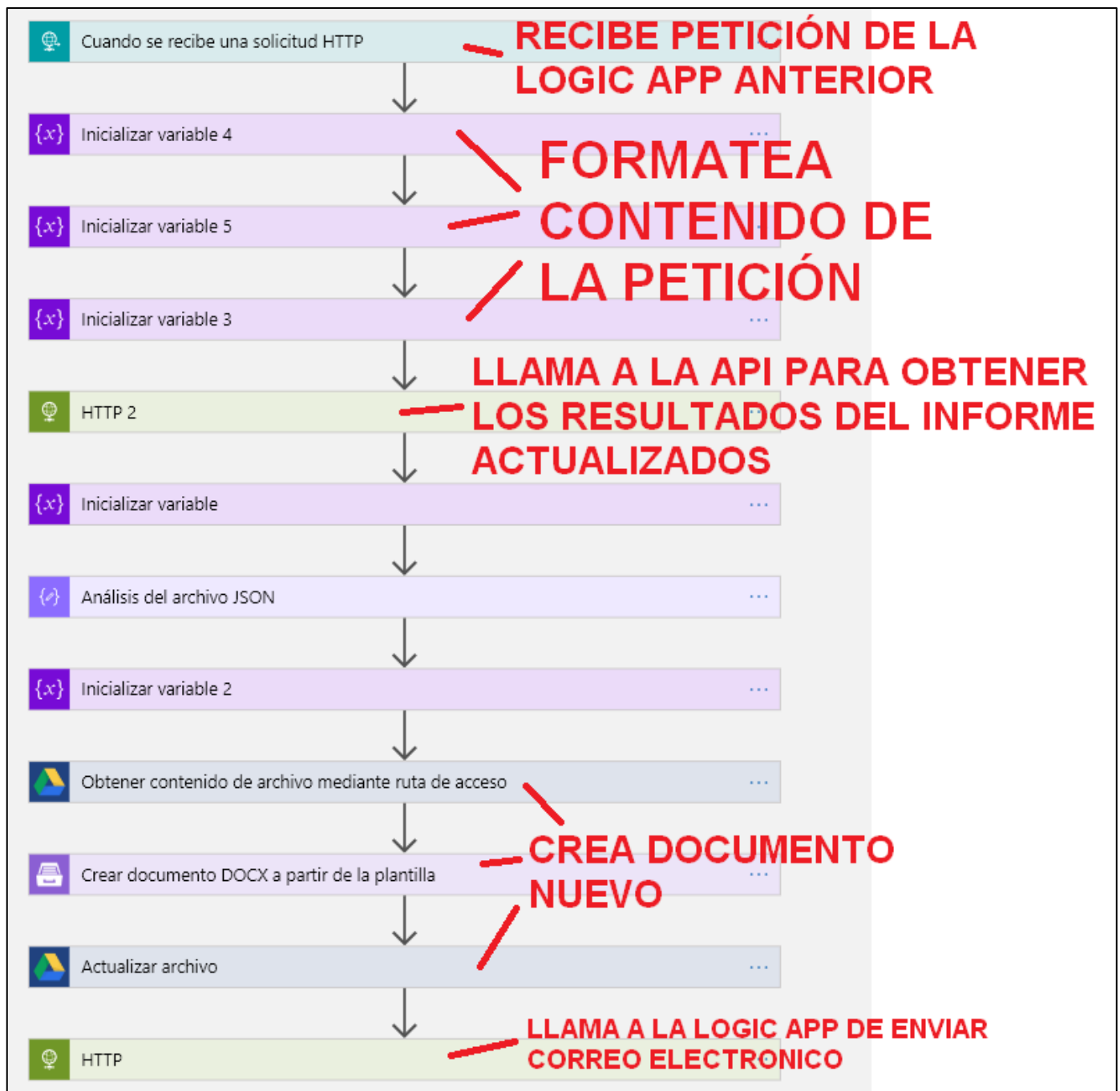


Figura 37. BuclInvoiceSummary

La última llamada HTTP del flujo se encarga de llamar a la última Logic App, encargada de enviar el correo electrónico.

EnviarCorreo

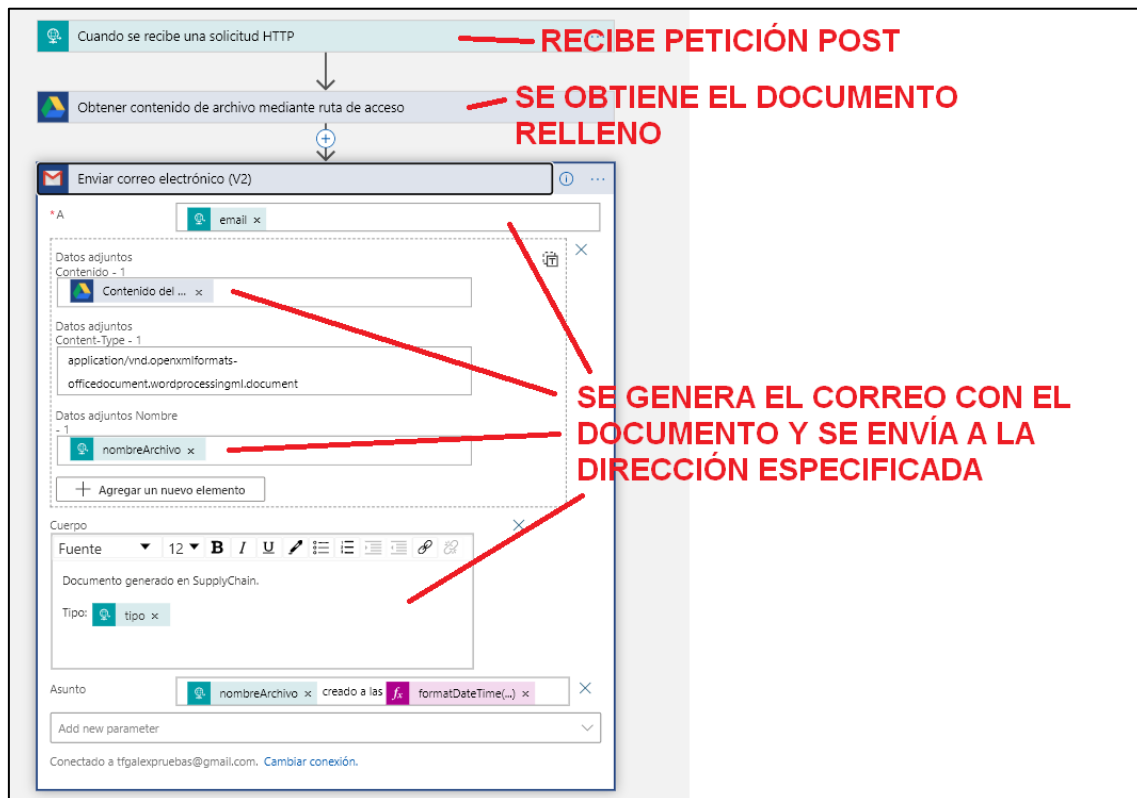


Figura 38. BucleInvoiceSummary

Como se muestra en la figura anterior, este flujo solo se encarga de enviar el correo electrónico.

5.7.2 Resultados

Para mostrar los resultados obtenidos se mostrarán capturas de pantalla de todo el proceso desde que se aplican los filtros hasta que se recibe el correo electrónico con el informe.

GESTIÓN

OPERACIONES

INFORMES

Informe de Facturas

Filtros

Fecha de factura desde Fecha de factura hasta

Id de factura

Facility comprador

Precio mayor/igual a Moneda

Comprador en EU ☒

Documento del informe

Acciones a realizar ☒ Nada ☐ Actualizar documento

Email

Intervalo de actualizacion

Se aplican filtros de fecha, moneda y precio neto

Cancelar

Aplicar Filtros

Figura 39. Aplicar filtros

Documento del informe

Acciones a realizar ☐ Nada ☒ Actualizar documento

Email

Intervalo de actualizacion

Se rellenan los datos para obtener el informe en el correo electronico

Resultados del filtrado

Mostrar

Filtrar contenido:

ID factura	Fecha factura	Precio	Moneda	Comprador europeo	FID comprador	Nombre del comprador	País	Ciudad	Dirección	CP	Seriales
888	26/06/2020	124	EUR	Si	Prueba1	Prueba1	España	Sabadell	Abcde, 2	08181	<div>Id : 15 Serial : testa1</div> <div>Id : 16 Serial : testa2</div>

Mostrando 1 a 1 de 1 resultados

Anterior Siguiente

Figura 40. Solicitud enviada

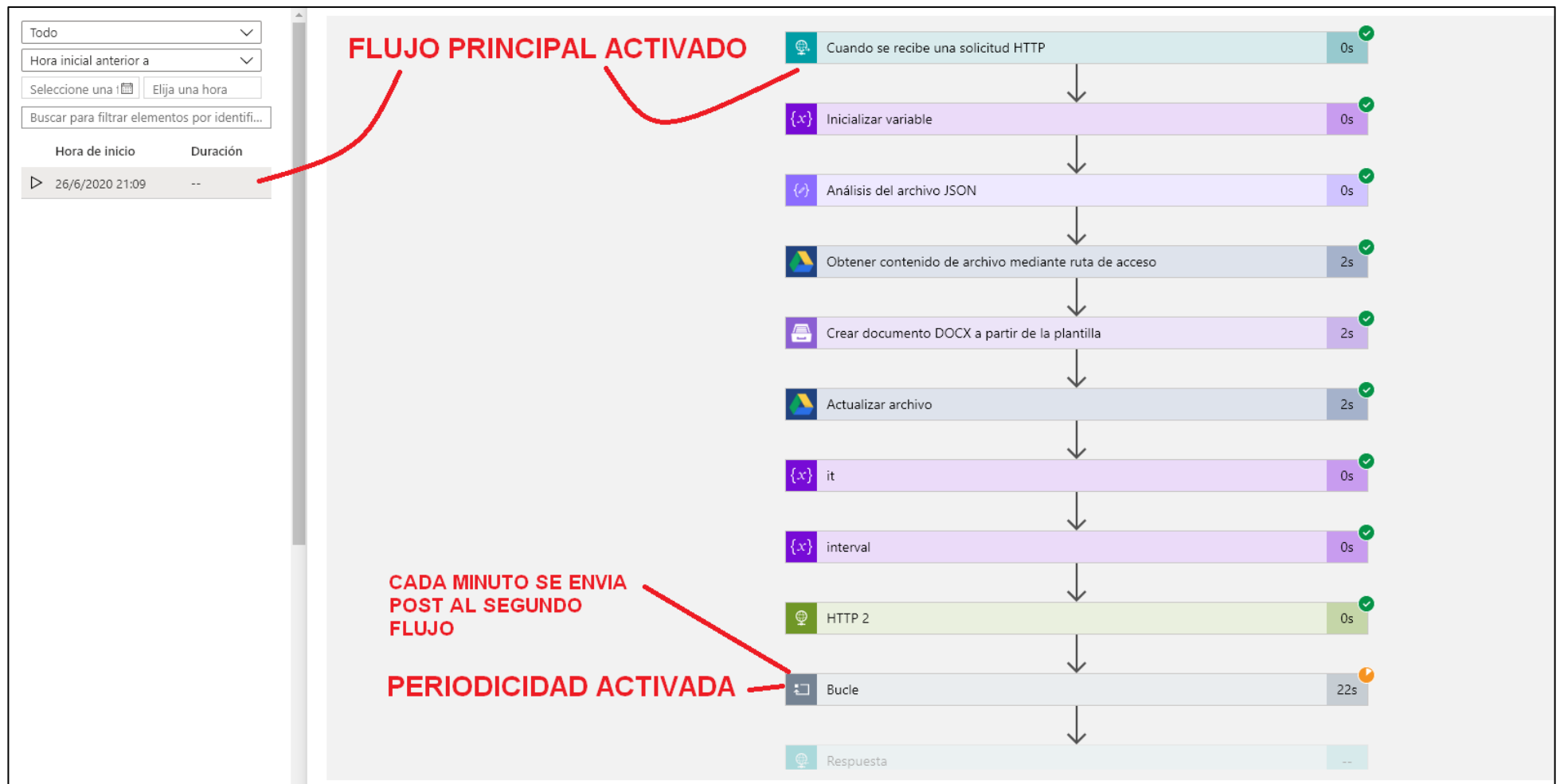


Figura 41. Solicitud enviada

[Inicio](#) > [BucleInvoiceSummary](#) > [Historial de ejecuciones](#) >

Historial de eje...

BucleInvoiceSummary

 Actualizar

Todo

Hora inicial anterior a

Seleccione una fecha

Elija una hora

Buscar para filtrar elementos por identifi...

	Hora de inicio	Duración
✓	26/6/2020 21:11	4.96 segundos
✓	26/6/2020 21:10	5.04 segundos
✓	26/6/2020 21:09	5.1 segundos

Ejecución de aplicación lógica

08586084082975517634584565978CU169

 Detalles de ejecución  Volver a enviar  Cancelar

**EL FLUJO SE HA ACTIVADO
CADA MINUTO**

Figura 42. Flujo 2

1-3 de 3 < > Es ▾

Principal Social Promociones

Se recibe un correo cada minuto

<input type="checkbox"/> ☆ yo	InvoiceSummary.docx creado a las 26-06-2020 09:11:55 - Documento generado en SupplyChain. Tipo: Resumen de facturas filtradas	21:11
<input type="checkbox"/> ☆ yo	InvoiceSummary.docx creado a las 26-06-2020 09:10:55 - Documento generado en SupplyChain. Tipo: Resumen de facturas filtradas	21:10
<input type="checkbox"/> ☆ yo	InvoiceSummary.docx creado a las 26-06-2020 09:09:54 - Documento generado en SupplyChain. Tipo: Resumen de facturas filtradas	21:10

Figura 43. Correos recibidos



tfgalexpruebas@gmail.com

para mí ▾

Documento generado en SupplyChain.

Tipo: Resumen de facturas filtradas



← Responder

➡ Reenviar

Autoguardado ☐ InvoiceSummary.docx - Guardado en Este PC Alex Sanchez

Archivo Inicio Insertar Diseño Disposición Referencia Correo Revisar Vista Programa Ayuda Diseño Disposición Buscar

viernes, 26 de junio de 2020 9:22

supply chain

RESUMEN DE FACTURAS FILTRADO

FACTURA ID	FECHA DE FACTURACIÓN	NOMBRE DEL COMPRADOR	PAÍS	CIUDAD	DIRECCIÓN	CÓDIGO POSTAL	MONEDA	PRECIO NETO
888	26/06/2020 0:00:00	Prueba1	España	Sabadell	Alcde. 2	08181	EUR	124

PRODUCTOS

FACTURA ID	SERIALES
888	testa1
888	testa2

Figura 44. Documento generado

5.7.3 Listado Logic Apps

Con la ejecución anterior se muestra el flujo completo para las facturas y la generación de sus informes. De la misma manera y como se muestra en la siguiente figura, todas las entidades de la plataforma web funcionan de forma similar pero utilizando sus llamadas a la API y uso de plantillas pertinentes.














Logic Apps 	
Universitat Autònoma de Barcelona (alumnes)	
+ Agregar ≡ Editar columnas ↺ Actualizar ↻ Probar la	
Suscripciones: Suscripción de Azure 1	
<input type="text" value="Filtrar por nombre..."/>	
10 elementos	
<input type="checkbox"/> Nombre 	Tipo 
<input type="checkbox"/>  BucleInvoiceSummary	Aplicación lógica
<input type="checkbox"/>  CreateDocArrivals	Aplicación lógica
<input type="checkbox"/>  CreateDocDispatches	Aplicación lógica
<input type="checkbox"/>  CreateDocInvoices	Aplicación lógica
<input type="checkbox"/>  CreateDocSerials	Aplicación lógica
<input type="checkbox"/>  EnviarCorreo	Aplicación lógica
<input type="checkbox"/>  UpdateDocArrivals	Aplicación lógica
<input type="checkbox"/>  UpdateDocDispatches	Aplicación lógica
<input type="checkbox"/>  UpdateDocInvoices	Aplicación lógica
<input type="checkbox"/>  UpdateDocSerials	Aplicación lógica

Figura 45. Listado Logic Apps

5.7.4 Monitorización y testeo

Tal y como se ha comentado en el apartado de la API, la parte de testeo ha consistido en hacer pruebas de usuario y manuales, aprovechando la interfaz de monitorización que ofrece Azure y permite detectar los errores de las ejecuciones, tanto de la plataforma web, de la API y de Logic Apps. Un ejemplo es el siguiente, que la ejecución de la Logic App se detiene y avisa que la petición es incorrecta. De esta forma sabemos que parte de la ejecución revisar de una forma casi automatizada.



Figura 46. Monitorización errores.

Para la monitorización más específica, se puede acceder a la parte de supervisión. Allí se observan varias gráficas personalizables. En este caso, muestran el resultado de la ejecución durante las últimas 3 horas, las ejecuciones facturables y la latencia de éstas.

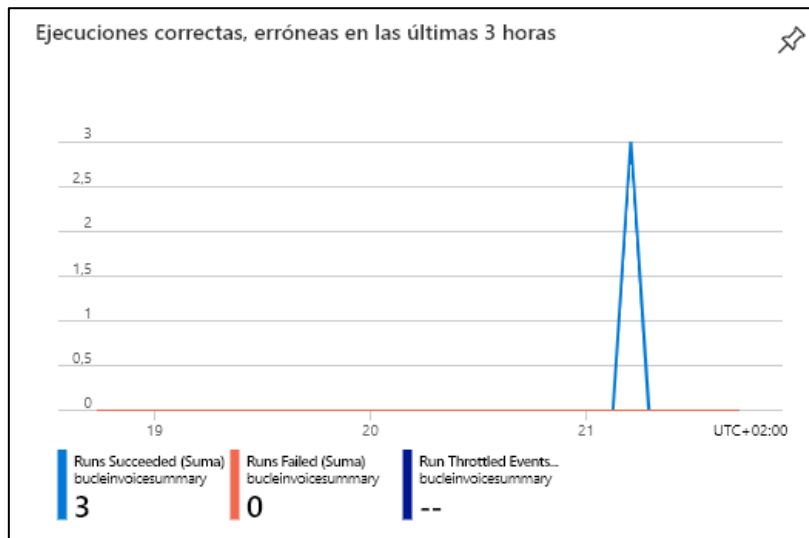


Figura 47. Gráfico 1

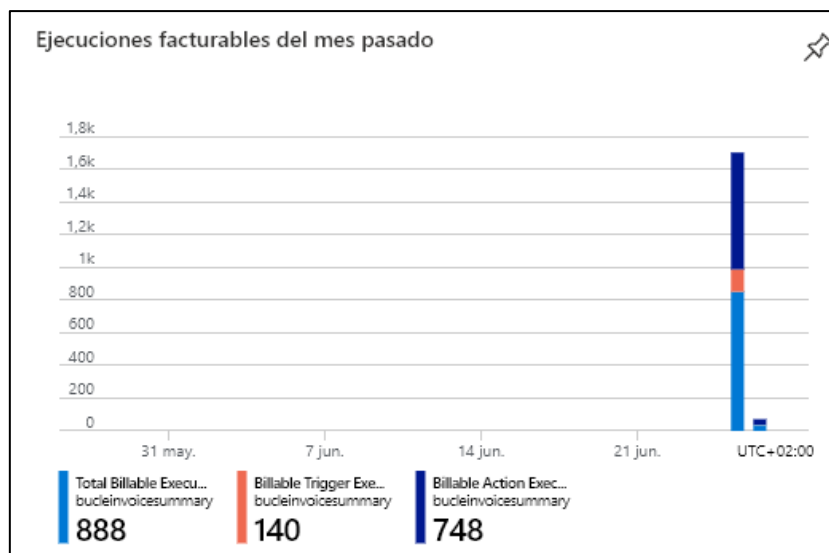


Figura 48. Gráfico 2

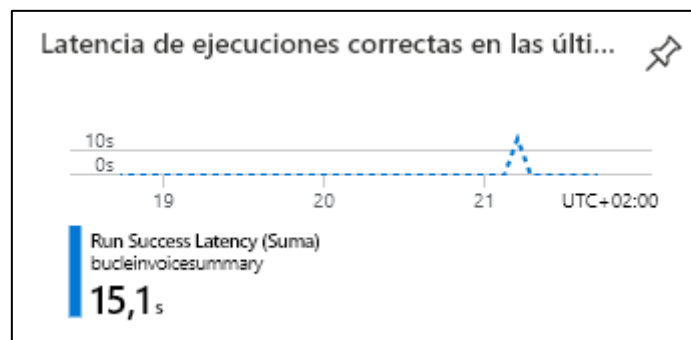


Figura 49. Gráfico 3

6. Presupuesto

Después de la implementación, puesta en producción y posterior ejecución y pruebas, se ha previsto un gasto aproximado del proyecto en cuestiones de mantenimiento. A través de la herramienta de análisis de costos se ha estimado que el conjunto de todos los recursos implementados suma una cantidad de 72.02€. La previsión al mes es de 80€.

Esta previsión incluye:

- Logic App
- Plan de suscripción para las API
- Base de datos
- App Service

El costo desglosado queda de la siguiente manera:



Recurso	Resource type	Cost  
> asp-tfgalex-b827	Plan de App Service	€35.79
> asp-tfgalex-a0df	Plan de App Service	€35.79
> enviarcorreo	Aplicación lógica	€0.17
> tfg-getinvoicesreport	Aplicación lógica	€0.12
> updatedocinvoices	Aplicación lógica	€0.07
> asp-tfgalex-bd0a	Plan de App Service	€0.05
> bucleinvoicesummary	Aplicación lógica	€0.03
> asp-tfgalex-ae7d	Plan de App Service	€0.03
> updatedocarrivals	Aplicación lógica	€0.01
> createdocarrivals	Aplicación lógica	<€0.01
> getinvoicesreportbypost	Aplicación lógica	<€0.01
> updatedocserials	Aplicación lógica	<€0.01
> updatedocdispatches	Aplicación lógica	<€0.01
> createdocinvoices	Aplicación lógica	<€0.01
> createdocserials	Aplicación lógica	<€0.01
> createdocdispatches	Aplicación lógica	<€0.01
> apitfgalex	Application Insights	€0
> tfgalex / tfgalex	SQL Server	€0
> apitfgalex	App Service	€0

Figura 50. Costos

Observando la tabla anterior, se debe destacar que el mayor coste proviene de la contratación del servicio para cada App Service por 35€ cada uno, que puede variar considerablemente en una situación en la que se necesita que accedan varios usuarios.

La base de datos y su servidor suponen 0€ de gasto puesto que viene contratadas con un servicio básico.

Las Logic Apps, por otro lado suponen un coste muy bajo puesto que suponen un coste por uso, y como se ha comentado anteriormente, solo ha accedido a la aplicación un usuario de forma concurrente. En un caso que se activaran las suscripciones y accedieran miles de usuarios, este precio podría aumentar considerablemente. Aun así, siguen siendo una opción muy eficiente para este tipo de aplicaciones.

7. Planificación

La fecha de finalización del proyecto ha sido marcada para finales de junio.

Es decir, para entonces debe haber una solución funcional y testeada del conjunto del proyecto.

Para cumplir con los términos marcados, el proyecto se ha dividido en dos partes:

- Fase de desarrollo de Plataforma Web y API
- Fase de automatización de procesos y monitorización con Logic Apps.

La primera de ellas ha sido marcada con un deadline **el 25 de mayo**.

La segunda parte debe estar finalizada un mes después, **el 15 de junio**.

De esta manera se dispondrá de dos semanas, aproximadamente, para hacer las pruebas finales de rendimiento.

Las últimas semanas antes de la entrega final servirán para finalizar el informe final, realizar el artículo y preparar la exposición.

7.1 Inicio del Proyecto

La primera fase del proyecto es la responsable de dar una idea concisa del trabajo a realizar, valorar las capacidades y límites del proyecto y dar una visión concreta del rumbo a seguir.

Fecha final: 08-03-2020

7.2 Captación de requisitos

Para captar los requisitos, se tendrá que realizar una investigación para concretar y responder varias preguntas como por ejemplo: qué necesita la plataforma web, cómo son las plataformas web similares, qué tipo de API se necesitará, qué tipo y cuántos recursos habrá que contratar en Azure, qué modelo de programación se seguirá, etc.

Fecha final: 13-03-2020

7.3 Preparación del entorno y diseño

Una vez respondidas casi todas las preguntas anteriores o todas las imprescindibles, hay que decidir cómo será la plataforma web, qué tipo de diseño tendrá, ...

Fecha final: 21-03-2020

7.4 Desarrollo y Test

Se desarrollará mediante TDD, es decir en todo momento se deberá probar que lo que se desarrolla cumplirá con las especificaciones expuestas.

Para ello, en primer lugar habrá que crear el entorno web, y, paralelamente, crear la API responsable de realizar las acciones.

Por último, se abordará el tema principal del proyecto, las Logic Apps.

Fecha final: 30-05-2020

7.5 Test y rendimiento

Para finalizar el desarrollo, habrá que realizar las pruebas de rendimiento finales para saber si la plataforma web es eficiente, en tiempo y consumo.

Puesto que la plataforma es una simulación de un caso real en el que deberemos de acceder a miles de datos, habrá que tener especial cuidado con las consultas a las bases de datos y controlar los tiempos.

Fecha final: 26-06-2020

7.6 Reorganización de la planificación

Ante la gran carga de trabajo supuesta en la primera parte del proyecto (estructuración y codificación de la plataforma web) se ha decidido realizar una reorganización de los plazos de la planificación además de una simplificación de las siguientes tareas:

- Las vistas de informes de pagos, órdenes, entregas y seriales quedan comentadas para reducir la carga de trabajo.
- La gestión de usuario se ha colocado en tarea opcional.
- La monitorización de errores se ha establecido a su vez como tarea supletoria.
- La Localización de la plataforma web (poder cambiar de idioma) queda marcada como tarea opcional.

Después de la replanificación, la fase uno del proyecto queda establecida para su finalización antes de junio.

La segunda fase (API y Logic Apps) se alarga dos semanas para poder hacer frente a las tareas de la fase uno de forma eficiente.

7.7 Trello

Para tener una organización concreta y saber qué tareas se deben completar, cuáles se han dado por cerradas, etc. se ha trabajado con una pizarra de Trello[12] que representa el backlog del proyecto con todas sus tareas.

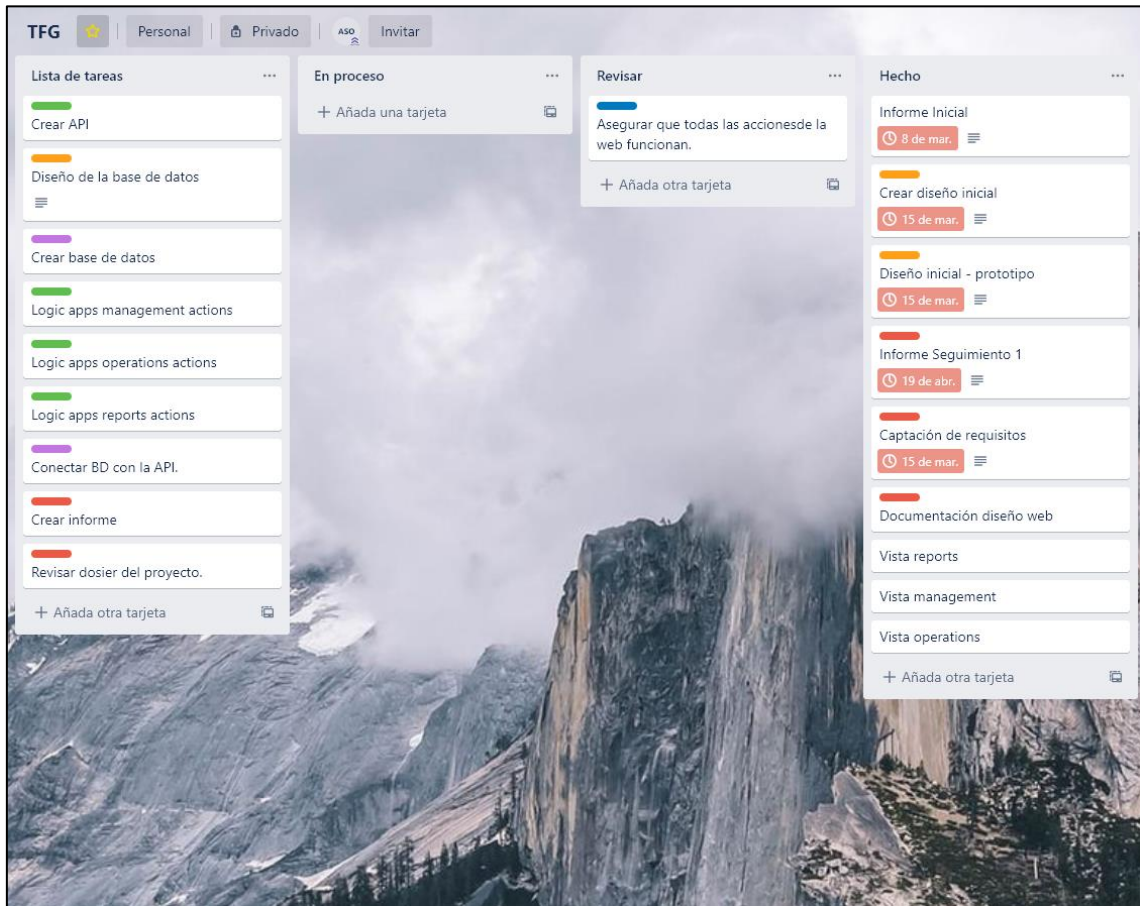


Figura 51: Trello

8. Conclusiones

Como ya se especificó al inicio del documento, este proyecto no ha sido solo un Trabajo de Fin de Grado, ha sido un desarrollo de software analizado, diseñado e implementado siguiendo los pasos del software integrado de una empresa real. El proceso de automatización de procesos en las llamadas a la API es la motivación por la cual se han desencadenado la consecución de tareas por cada etapa de la vida del software.

La fase de planificación del proyecto, una etapa clave para la gestión de los tiempos, se realizó en la fase más temprana y se ha ido actualizando a la vez que lo hacía el proyecto. Por un lado, el tiempo estipulado del proyecto en un inicio no se sopesó con total certeza debido a la falta de experiencia y a la situación extraordinaria vivida durante su desarrollo. Por otro lado, se aplicaron los errores de cálculo a tiempo para poder obtener un resultado funcional y lo más eficiente posible:

- Durante algunas fases, el tiempo calculado para algunas tareas fue muy optimista. Concretamente en la codificación del front-end de la aplicación.
- Las etapas del ciclo de vida del software no se siguieron totalmente de forma cronológica como se representa en este documento. Durante el desarrollo se tuvo que volver a la fase de diseño más de una vez para corregir ambigüedades y errores. La fase de requisitos ha sido modificada sobretodo en la fase de desarrollo del front-end, para ajustarse al resultado que realmente se requería.

Como último aspecto, me gustaría hablar en primera persona y expresar lo que ha significado el desarrollo de este proyecto. Gracias a este trabajo y durante estos meses, mi interés en relación con el Cloud Computing ha ido creciendo hasta el punto de descubrir un conjunto de recursos y servicios disponibles para gestionar proyectos, monitorizar su desarrollo y ejecución y varias aplicaciones futuras para todos ellos. En definitiva, un gran abanico de posibilidades para proyectos futuros.

Para mí el proyecto ha supuesto un primer contacto en el deploy de una plataforma web con su API, la gestión y automatización de la integración continua, la monitorización de su ejecución, etc. En cuanto al desarrollo, me ha servido para poner a prueba los conocimientos obtenidos a lo largo del grado y la especialización en Ingeniería del Software, como el control de versiones, requisitos del software, documentación, gestión y administración de bases de datos, etc.

Como puntos negativos que me gustaría destacar, la planificación tan optimista dada en un inicio ha supuesto una piedra en el camino que ha provocado la reorganización de las tareas a realizar y abstracción del trabajo. La carga de trabajo manejada en un inicio no se valoró correctamente hasta llegar a un punto muy avanzado en el desarrollo.

Como evaluación personal, considero que el trabajo ha sido bueno y la motivación llevada durante estos meses ha significado un gran punto a favor. Como aspecto de aprendizaje, personalmente ha significado un gran salto para mí.

8.1 Puntos para mejorar

Finalmente se hace un balance de las líneas de trabajo futuro en las que se incluyen los siguientes puntos:

- Se debe destacar la necesidad de implementación de test, que durante todo el proyecto ha sido manual y user testing.
- La finalización de algunos componentes, que finalmente no se han desarrollado para reducir la carga de trabajo, también es un aspecto para revisar:
 - Introducir sesiones de usuario y autenticación.
 - Localización para traducir los textos del cliente, es decir un sistema de traducción.
 - Permitir una total administración al usuario en cuanto a las suscripciones a las Logic Apps
 - Asegurar *responsive* web, que hasta el final no ha significado un punto para tener en cuenta, ya que la plataforma web era solamente un punto de inicio para empezar a trabajar con la API y los servicios cloud como Logic Apps. Es decir, no se pretendía desarrollar una página web que atraiga nuevos usuarios, sino que una que sea lo más funcional y eficiente posible para los reales usuarios, empresas relacionadas con logística, que necesita registrar facturas, envíos, etc.
- Valorar alternativas a Microsoft Azure. No por muy buenos resultados que haya dado esta elección significa que sea la mejor de todas. Un estudio de alternativas que ofrezcan cloud computing puede significar un manejo más amplio y una capacidad de desarrollo mayor.

9. Bibliografía

- Portal de Microsoft Azure [1]
<https://portal.azure.com/#home>
- Información acerca de Azure [2]
<https://azure.microsoft.com/es-es/services/logic-apps/>
- Documentación Logic Apps [3]
<https://docs.microsoft.com/es-es/azure/logic-apps/logic-apps-overview>
- Metodología Waterfall vs Agile [4]
<https://thedigitalprojectmanager.com/es/agile-frente-a-waterfall/>
<https://www2.deloitte.com/es/es/pages/technology/articles/waterfall-vs-agile.html>
<https://www.esan.edu.pe/apuntes-empresariales/2017/05/proyectos-en-metodologia-waterfall-o-agile-hacia-donde-vamos/>
- Información de Metodología Waterfall [5]
<http://design-toolkit.recursos.uoc.edu/es/waterfall/>
[Recursos de la asignatura de Gestión de Proyectos e Ingeniería del Software]
- Gestión de recursos [6]
<https://www.wrike.com/es/blog/que-es-la-gestion-de-recursos-y-por-que-es-importante/>
<https://www.ceupe.com/blog/la-gestion-de-los-recursos.html>
- Partes del diseño [7]
https://es.wikipedia.org/wiki/Dise%C3%B1o_web
<https://neoattack.com/neowiki/disenio-web/>
- Wireframes, Mockups y Prototipos [8]
<https://medium.com/rocket-studio-ux/wireframe-mockup-y-prototipos-en-busca-de-sus-diferencias-23a03bcbdb69>

- Wireframes [9]
<https://webdesdecero.com/wireframes-que-son-y-como-crearlos/>
- Herramientas de prototipado [10]
<https://webdesdecero.com/wireframes-que-son-y-como-crearlos/>
<https://www.axure.com/>
<https://docs.axure.com/axure-rp/reference/getting-started-video/>
<https://neoattack.com/proyectos/>
- DDD [11]
<https://medium.com/@jonathanloscalzo/domain-driven-design-principios-beneficios-y-elementos-primera-parte-aad90f30aa35>
<https://devexperto.com/domain-driven-design-1>
- Trello [12]
<http://trello.com>
- Creación de la base de datos [13]
<https://portal.azure.com/#create/Microsoft.SQLDatabase>
<https://www.youtube.com/watch?v=QrCa25p4ixE>
- Diagrama de la base de datos
<https://www.youtube.com/watch?v=KECvDwc9E3A>
- Creación de la API web con ASP.NET Core
<https://medium.com/@azaharafernandezguizan/8-pasos-para-crear-una-web-api-con-net-core-25323708cac0>
<https://docs.microsoft.com/es-es/aspnet/core/tutorials/first-web-api?view=aspnetcore-3.1&tabs=visual-studio>
- Inyección de dependencias [14]
<https://www.arquitecturajava.com/el-patron-de-inyeccion-de-dependencia/>

- Principios SOLID [15]
[https://profile.es/blog/principios-solid-desarrollo-software-calidad/#:~:text=Los%205%20principios%20SOLID%20de,%E2%80%93%20Liskov%20Substitution%20Principle%20\(LSP\)](https://profile.es/blog/principios-solid-desarrollo-software-calidad/#:~:text=Los%205%20principios%20SOLID%20de,%E2%80%93%20Liskov%20Substitution%20Principle%20(LSP))
- Mediator [16]
<https://docs.microsoft.com/es-es/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/microservice-application-layer-implementation-web-api>
<https://www.programmingwithwolfgang.com/mediator-pattern-in-asp-net-core-3-1/>
- POSTMAN [17]
<https://profesores.virtual.uniandes.edu.co/~isis2603/dokuwiki/doku.php?id=tutoriales:postman>
<https://www.postman.com/>
- Esquema Flujo BackEnd [18]
<https://app.diagrams.net/>
- Logic Apps [19]
<https://docs.microsoft.com/es-es/azure/logic-apps/>
- App Service[20]
<https://azure.microsoft.com/es-es/services/app-service/>
<https://channel9.msdn.com/Blogs/msmdotnet/Qu-es-App-Service>
- Crear documentos con plantillas y Visual Basic con PLUMSAIL [21]
<https://plumsail.com/docs/documents/v1.x/search.html?q=logica+apps+word#>

10. Figuras

- Figura 1. Microsoft Azure Services
- Figura 2. Interfaz Logic Apps
- *Figura 3. Flujo de trabajo de ejemplo*
- *Figura 4. Flujo de trabajo ampliado de ejemplo*
- *Figura 5. Wireframe página principal*
- *Figura 6. Wireframe informe de facturas*
- *Figura 7. Interfaz de Axure*
- *Figura 8. Página principal*
- *Figura 9. Menú lateral Axure*
- *Figura 10. Resultado de la traducción en la web*
- *Figura 11. Interacciones de los botones Español-English*
- *Figuras 12: Vistas del diseño*
- *Figura 13. Creación recurso BD en Azure.*
- *Figura 14. Interfaz Microsoft SQL Server Management Studio.*
- *Figura 15. Diagrama de la base de datos.*
- *Figura 16. Esquema de funcionamiento.*
- *Figura 17. Organización end-points.*
- *Figura 18. Ejemplo inyección de dependencia.*
- *Figura 19. Esquema Mediator DDD*
- *Figura 20. Esquema Mediator DDD Arrival Summary[18]*
- *Figura 21. Arrival API*
- *Figura 22. Handler de Arrival*
- *Figura 23. Api en ejecución*
- *Figura 24. Uri y parámetros*
- *Figura 25. Respuesta API*

- *Figura 26. URI, Body POST*
- *Figura 27. API Response*
- *Figura 28. Base de datos Arrivals.*
- *Figura 29. Distribución código Front-end*
- *Figura 30. Servicio para obtener los Arrivals.*
- *Figura 31. Conjunto de endpoints del Front-end*
- *Figura 32. App Services*
- *Figura 33. Información App Service frontend*
- *Figura 34. Implementaciones App Service.*
- *Figura 35. UpdateDocInvoices 1/2*
- *Figura 36. UpdateDocInvoices 2/2*
- *Figura 37. BucleInvoiceSummary*
- *Figura 38. BucleInvoiceSummary*
- *Figura 39. Aplicar filtros*
- *Figura 40. Solicitud enviada*
- *Figura 41. Solicitud enviada*
- *Figura 42. Flujo 2*
- *Figura 43. Correos recibidos*
- *Figura 44. Documento generado*
- *Figura 45. Listado Logic Apps*
- *Figura 46. Monitorización errores.*
- *Figura 47. Gráfico 1*
- *Figura 48. Gráfico 2*
- *Figura 49. Gráfico 3*
- *Figura 50. Costos*
- *Figura 51: Trello*

11. Tablas

11.1 Tablas de requisitos funcionales

- *Tabla 1: RQF – 01*
- *Tabla 2: RQF – 02*
- *Tabla 3: RQF – 03*
- *Tabla 4: RQF – 04*
- *Tabla 5: RQF – 05*
- *Tabla 6: RQF – 06*
- *Tabla 7: RQF – 07*
- *Tabla 8: RQF – 08*
- *Tabla 9: RQF – 09*
- *Tabla 10: RQF – 10*
- *Tabla 11: RQF – 11*
- *Tabla 12: RQF – 12*
- *Tabla 13: RQF – 13*
- *Tabla 14: RQF – 14*
- *Tabla 15: RQF – 15*
- *Tabla 16: RQF – 16*

11.2 Tablas de requisitos de datos

- *Tabla 17: RQD – 01*
- *Tabla 18: RQD – 02*
- *Tabla 19: RQD – 03*
- *Tabla 20: RQD – 04*
- *Tabla 21: RQD – 05*

- *Tabla 22: RQD – 06*
- *Tabla 23: RQD – 07*
- *Tabla 24: RQD – 08*
- *Tabla 25: RQD – 09*
- *Tabla 26: RQD – 10*

11.3 Tablas de requisitos de datos

- *Tabla 27: RQS – 01*