# Álvaro Maleno Alférez.

1. Introducción

Se puede definir un proyecto como la “Idea de una cosa que se piensa hacer y para la cual se establece un modo determinado y un conjunto de medios necesarios”. Si hablamos de este proyecto en particular, lo hacemos acerca de realizar una aplicación de consulta de datos personalizados y en tiempo real, acerca del Covid - 19 . Los medios, que serán dilucidados a medida que avance la redacción del mismo, no están exentos de discusión.

Como todos sabemos, existe un riesgo real de contagio, y una posibilidad de evitarlo estando bien informados. Los datos ofrecidos por las instituciones con o sin ánimo de lucro y aquellos puestos a disposición por el gobierno requieren de un arduo proceso de, primero filtrado y posteriormente tratamiento para ser dispuestos de manera entendible y útil a pesar de que el criterio de transparencia dicta su labor.

Este proceso no suele estar a disposición de todos los ciudadanos, pero, con la ayuda de un desarrollador, puede estarlo fácilmente. Esa es la finalidad última de este proyecto.

## 1.1 Descripción

La necesidad de estar informados de manera transparente y en tiempo real acerca de lo que sucede en un mundo globalizado e interconectado y la reciente transmisión de una pandemia a nivel mundial hace necesaria una solución abierta, sencilla y ágil para hacer llegar la aplicación a toda la ciudadanía.

La información de fácil acceso, fiable, inmediata y concreta se ha convertido en un bien imprescindible para el desarrollo de nuestras vidas. No se trata solamente de que, como ciudadanos de un sistema democrático tengamos la obligación de estar informados con la finalidad de ejercer nuestros derechos y poderes otorgados de manera responsable y cívica, también nos resulta imprescindible a la hora de evaluar la actuación de nuestros gobernantes.

Pero hay más. Buena parte de las decisiones que tomamos en nuestra vida diaria están basadas - o nos gustaría que lo estuviesen - en datos concretos y significativos que nos dan una idea de la realidad. Por ejemplo, a nadie se le ocurriría escoger como lugar de vacaciones la ciudad en la que acaba de estallar una guerra civil. Tampoco nos gustaría arriesgar la salud yendo a un país que está siendo sacudido por una pandemia importante. De la misma manera planificamos la cantidad monetaria destinada al ahorro que estableceremos en función de nuestras previsiones económicas.

La falta de tiempo y la marcha acelerada de la vida moderna provoca, si no en todas las ocasiones, al menos en la mayoría, que los momentos para realizar determinada tarea sean escasos. Necesitamos, entonces, que todas las herramientas que empleemos para llevar estas tareas a término estén disponibles en cualquier momento, es decir, justo cuando las requerimos. Cuando la herramienta es la información nos surge una necesidad de la misma.

En este sentido, una simple aplicación que recoja los datos estadísticos de los distintos organismos gubernamentales y no gubernamentales y los muestre de forma nítida y precisa, así como entendible tiene un amplio público al que dirigirse y constituye un proyecto más que atractivo.

Se trata de ofrecerle al mundo los datos necesarios en tiempo real acerca de la marcha de la pandemia de forma personalizada a otras aplicaciones que decidan conectarse.

Estaríamos, entonces, hablando de un servicio API Rest, consumible mediante cuenta de usuario que ofrecería una serie de datos estadísticos actualizados y bajo demanda.

## 1.2 Objetivos

### 1.2.1 Objetivos generales:

* Desplegar un servicio REST API.
* Consumir los datos que ofrecen los organismos oficiales acerca del Covid-19.
* Servir esos datos fiablemente y de manera fácilmente entendible.
* Poner a disposición de otras aplicaciones el conjunto de datos relativos a la evolución, pasada y presente de la pandemia ocasionada por el Covid-19 en todo momento.

### 1.2.2 Objetivos Específicos:

* Acceder a la información y a las operaciones que puedan derivarse de sus datos.
* Ofrecer la media de fallecidos por infectado durante un período determinado.
* Mostrar la suma de fallecidos dentro de unas fecha específica.
* Consultar los datos relativos a la pandemia, especialmente si eres un desarrollador.
* Ofrecer consultas dentro de unas fechas determinadas.
* Consultar para un lugar geográfico concreto.

## 1.3 Justificación

Nuestra toma de decisiones diaria depende, en gran medida de la información que recibimos acerca del mundo que nos rodea. Nuestras próximas vacaciones, qué seguridad podemos tener acerca de salir o no al supermercado y un sinfín más de pequeñas y grandes preguntas cuya respuesta depende, en gran medida del estado de la pandemia global surgida a raíz del Covid - 19 son ejemplos de ello.

Como los organismos oficiales ofrecen únicamente los datos en bruto y sin filtrar u ordenar de manera fácilmente visualizable y, los medios de comunicación cumplen con su función más o menos bien, pero no pueden garantizar nuestra necesidad de información en todo momento, se hace necesario un servicio que focalice y cubra las demandas de datos de una forma más personalizada de lo que actualmente se está haciendo. Surge, así, un motivo que de sobra justifica la realización de un servicio API que proporcione los datos sencilla y adecuadamente.

## 1.4 Beneficiarios del proyecto

En un principio cualquier desarrollador o empresa que necesite consultar los datos en tiempo real - empresas del sector turístico, creadores de aplicaciones y páginas web que prefieran delegar el tratamiento de los datos, etc.. - son los principales beneficiarios del proyecto.

No obstante, la amplitud del mismo, el hecho de que favorece la aparición de aplicaciones que consuman y ofrezcan los datos y la necesidad generalizada de la información convierten a la ciudadanía global en principales beneficiarios del proyecto.

## 1.5 Información adicional

### 1.5.1 Metodología del proyecto

En el terreno de las empresas tecnológicas desde tiempo atrás ha venido implementándose, ganando cada vez más fuerza, el empleo de las conocidas metodologías ágiles. Fue en el año 2001, con la declaración del manifiesto ágil por parte de un grupo de desarrolladores de elevada influencia internacional cuando comenzaron a surgir toda una serie de métodos nuevos de desarrollo enfocados principalmente en la relación con el cliente y la adaptación al cambio frente al desarrollo exhaustivo de documentación.

Una de las consecuencias es el desarrollo por iteración. Es decir, se comienza por desarrollar una parte pequeña de la aplicación - a veces llamado método bola de nieve - y, si cumple con lo requerido se pasa a la siguiente fase de desarrollo, haciendo crecer la aplicación. En caso contrario, se realizaría una segunda iteración sobre la parte desarrollada. Como curiosidad se puede comentar que este método ha comenzado a seguirse para la elaboración de novelas, libros de texto y otras muchas actividades.

Obviamente esto no sustituye el tradicional estilo en cascada, comenzando por la fase de análisis, diseño, implementación, pruebas y puesta en marcha, pero hace la fase de implementación mucho más flexible y llevadera.

La división del trabajo en tareas pequeñas y fácilmente asumibles por los desarrolladores y los analistas de calidad de las aplicaciones que compone el método Kanban y su control semanal o diaria a través de reuniones cortas que ayudan a focalizar el trabajo y mantener claras unas prioridades hacen del método Scrum un gran aliado.

Puesto que se buscará seguir una filosofía más cercana a la responsabilidad individual la prueba del programa la realizarán los propios desarrolladores.

En principio, siguiendo el desarrollo en cascada, este proyecto empleará todas las técnicas puestas a disposición por las metodologías ágiles de desarrollo de programas informáticos.

### 1.5.2 Conocimientos

Se pueden clasificar en conocimientos técnicos y conocimientos de gestión. En el plano de los conocimientos técnicos, teniendo en cuenta que es necesario, para el desarrollo de una aplicación, completar diferentes fases interconectadas, serán necesarios conocimientos relativos al área de la programación informática.

El análisis y la identificación de problemas y requerimientos funcionales de una aplicación, aquellas funciones y tareas que debe de realizar una aplicación para resolver un problema determinado, suponen una base de conocimiento importante y muy necesario. Saber diseñar la aplicación para que sea eficiente y fácilmente mantenible sería el siguiente paso lógico. Por último, el desarrollo dentro de diversas tecnologías y lenguajes de programación es, también, un requerimiento de conocimiento básico.

Esta formación ha sido completada mediante el estudio de un ciclo superior en Desarrollo de Aplicaciones Multiplataforma y, extendida, en el caso relacionado con el lenguaje de programación a emplear con formación laboral y cursos extensivos de plataformas en en línea.

Con respecto a los conocimientos de gestión demandados, todos relacionados con el empleo de metodologías ágiles y con la correcta planificación de los tiempos y su diligente realización, hay que mencionar que es necesaria formación adicional satisfecha de dos maneras fundamentalmente: estudio y lectura de documentación y formación laboral en el puesto de trabajo.

### 1.5.3 Legislación

Si bien no existe una legislación específica al respecto, pues la transmisión de información se encuadraría dentro del derecho individual de expresión, la proliferación de noticias falsas y el permanente estado de desinformación en el que algunos colectivos se encuentran inmersos hizo que la Unión Europea convocase varios foros de discusión al respecto.

En ellos se decidió que, a pesar de que cerca de un 80% de la población tenía la percepción de que esta difusión de información falsa era perjudicial para los estados democráticos no se iba a regular. A pesar de ello se redactó un código de buenas prácticas sobre la desinformación para toda la UE entre, las que se incluye, la fidedignidad de la misma.

Siguiendo estas buenas prácticas, se establece como requisito que la recolección de datos únicamente proceda de fuentes oficiales, siendo éstas más o menos fiables dependiendo de sus criterios de evaluación.

2. Planificación temporal

La planificación se llevará a cabo siguiendo el modelo tradicional de desarrollo en cascada comenzando por el diseño, siguiendo por la implementación y finalizando con la realización de pruebas y su puesta en marcha. Todo el trabajo se realizará desde casa debido al decreto del estado de alarma.

## 2.1 Diseño

Un diseño comienza siempre con la tarea previa de análisis. Es necesario saber qué requisitos va a cumplir la aplicación para solucionar determinado problema y con qué funcionalidades va a alcanzar dicha solución. Es lo que se conoce como Análisis de Requerimientos y Análisis Funcional. Son necesarios, por tanto, dos perfiles: Analista y Arquitecto o Diseñador.

Para cada uno de los análisis se emplearán dos días. Uno para entender adecuadamente la tarea y otro para realizarla y corregirla en caso necesario.

Una vez completados los análisis, éstos serán plasmados en un diseño que posteriormente deberá ser implementado. Los diseños serán mínimos, para evitar que la tarea se extienda demasiado y estarán divididos en dos tipos: diseño de aplicación y diseño de base de datos.

Para que la aplicación sea fácilmente comprendida por cualquier desarrollador que la implemente y el que la mantenga, pero también por cualquier persona con conocimientos informáticos, pues, al tratarse de una aplicación pensada para realizar funcionalidades para otras aplicaciones, serán necesarios conocimientos técnicos, dos tipos de diagrama: un diagrama de flujo y uno de clases.

Como los requerimientos funcionales habrán sido previamente informados, el diagrama de flujo se ocupará de ponerlos en relación a los pasos que seguirá el algoritmo que compone la aplicación de una forma aún primitiva y esquemática. Para eso se empleará un día para realizar una primera propuesta, y otro para corregir y mejorar la misma.

Una vez conocido el flujo de la aplicación, se pueden identificar las clases y subclases junto con las operaciones de las que cada una se hará responsable y planificarlas en un diagrama de clases. De nuevo, serán necesarios dos días; uno para realizar una primera propuesta y otro para revisarla, corregirla y mejorarla en caso necesario.

Por otro lado, es necesario un buen diseño de la forma en la que se almacenarán los datos para evitar duplicados y garantizar un buen servicio de los mismos. Esta necesidad se cubrirá realizando un diagrama entidad relacional. Como la base de datos no será demasiado extensa, bastará un único día para desarrollar esta actividad.

## 2.2 Implementación

Una vez diseñada la aplicación, se pasaría a la fase de implementación. Puesto que es en esta fase en la que surgen la mayoría de los problemas asociados al desarrollo, el tiempo para su realización será mayor. Dos perfiles son necesarios: Desarrollador de base de datos y desarrollador de aplicaciones.

Lo primero a poner en funcionamiento será la persistencia de los datos. De nuevo, al no ser demasiado extensa, únicamente se emplearán cuatro días. Uno para generar la base de datos. Otro para ponerla en funcionamiento y dos adicionales para probarla intensivamente y comprobar que funciona. Si fuese necesario se corregirían los errores antes de pasar a la siguiente tarea. Esta tarea será llevada a cabo por el desarrollador responsable que realizará una serie de pruebas de funcionalidad.

Con la base de datos como respaldo se pasaría a desarrollar el API Rest generando, primero, las clases necesarias e implementando posteriormente los procesos que sean requeridos. Se estima que la primera tarea ocupará dos días y tres la segunda, pues necesita de mayor algoritmia.

Las pruebas de calidad y funcionalidades se realizarían durante un día completo junto con la subsanación de los errores. Posteriormente se desplegará la aplicación en un entorno distinto al de desarrollo para comprobar que funciona adecuadamente en más de un ordenador. Finalmente se pondrán en marcha nuevas pruebas a toda la aplicación en su conjunto para comprobar que, una vez desplegada, realizará su tarea adecuadamente. Para cada una de estas tareas se dedicará un día completo.

Siguiendo con el empleo de las metodologías ágiles, la aplicación se mostrará durante dos días a tantos usuarios como sea posible y, durante los cuatro siguientes, habiendo tomado debida cuenta de la opinión de los usuarios, se realizarán las modificaciones correspondientes.

## 2.3 Cumplimiento de tareas

A pesar de que se ha dado margen de maniobra para la subsanación de los errores durante el mismo desarrollo del proyecto los retrasos y el surgimiento de imprevistos es inevitable. Es, entonces, imprescindible el perfil del Gestor de Proyectos.

Son mencionables los producidos durante la fase de diseño, pues no siempre es sencillo captar y entender cada uno de los requerimientos necesarios a priori. Cabe destacar también que, durante la fase de implementación el diseño puede variar por el propio conocimiento disponible o, en este caso, por la necesidad de adecuar la base de datos a diferentes requerimientos que se han demostrado más eficaces durante el desarrollo de la misma, como la división en varias tablas de una de ellas o la inserción de un nuevo identificador para acelerar los procesos de búsqueda por fecha.

3. Análisis de la competencia

Si bien es cierto que el acceso a la información relativa al Covid-19 tiene un planteamiento transparente y de libre consulta, siendo provista por diferentes instituciones y medios gubernamentales, la práctica totalidad de los medios de consulta la ofrecen en bruto, segmentada o analizada estadísticamente sí, pero no tienen la posibilidad de obtenerla en el momento bajo criterios personalizados. Tampoco es posible acceder a la información de una medida estadística sin realizar uno mismo los cálculos tras consumir los datos.

Surge entonces un nicho de mercado muy preciso en el que la mínima competencia lo convierte en un sinfín de posibilidades. Puede no ser monetizable como tal, pero sirve de escaparate para la generación de otros recursos por los que diversos clientes pueden estar dispuestos a pagar, como, por ejemplo, el desarrollo de una API de consulta de datos de movimientos turísticos.

Si bien existen varias APis gratuitas como https://covid-19-apis.postman.com/ o API-nCov2019 ambas poseen la misma debilidad, no es posible consultar información para más de un país bajo demanda a la vez, como tampoco ofrece una manipulación estadística de los datos. La principal fortaleza de la aplicación desarrollada para este proyecto es, precisamente, que ofrece la posibilidad de obtener los datos relativos a uno o más países, dentro de un rango de fechas y con operaciones estadísticas relevantes para su lectura.

Como conclusión, se puede mencionar que no existe una competencia en firme para un servicio similar, más allá del que ofrecen organismos públicos y oficiales, que esta información no es personalizable a través de su demanda y, que el desarrollo de un proyecto semejante posibilita presentar una candidatura en firme al desarrollo de otros proyectos, tal vez más rentables.

## 3.1 <https://covid-19-apis.postman.com/>

Su documentación se encuentra accesible desde <https://documenter.getpostman.com/view/4074074/SzS7Pkup?version=latest>:

Desde ella se puede observar que únicamente permite dos operaciones: obtener el conjunto de datos para un país u obtenerlos de forma paginada para el conjunto global. Además, como se puede ver en el ejemplo inferior, estos datos no se encuentran desglosados por fecha, sino que son sumas totales:

{

"country": "Italy",

"country\_abbreviation": "IT",

"total\_cases": "41,035",

"new\_cases": "0",

"total\_deaths": "3,405",

"new\_deaths": "0",

"total\_recovered": "4,440",

"active\_cases": "33,190",

"serious\_critical": "2,498",

"cases\_per\_mill\_pop": "679.0",

"flag": "https://www.worldometers.info/img/flags/it-flag.gif"

}

Pero esta no es la única aplicación que tienen disponible. Desde el enlace a la siguiente documentación se pueden observar un conjunto mayor de operaciones: <https://documenter.getpostman.com/view/10808728/SzS8rjbc?version=latest>

En este caso se pueden recuperar datos por país y fecha, pero no por rango de fecha. Y, pese a que la cantidad de operaciones es muy elevada, pues hay recogidos datos para el conjunto de la población mundial por país y fecha, si quisiésemos hacernos una idea de la marcha de la enfermedad en Italia durante las dos últimas semanas necesitaríamos realizar al menos 14 peticiones vía https antes de disponer del conjunto de datos a analizar.

|  |  |  |
| --- | --- | --- |
| **Fortalezas API propia** | **Debilidades API Postman** | **Fortalezas API Postman** |
| Consulta por rango de fechas para un país. | Una consulta para un país por día. | Gran cantidad de información. |
| Una única petición para obtener toda la información. | Necesidad de realizar múltiples peticiones para obtener un mapa completo. | API muy eficiente. |

## 3.2 API-nCov2019

Accesible desde <https://apimarket.nubentos.com/store/apis/info?name=API-nCoV2019&version=1.0.0&provider=owner-AT-nubentos.com&tenant=nubentos.com> .

Este API posee únicamente cinco operaciones: obtener los casos para un día concreto, obtener los casos aún bajo sospecha, casos confirmados, muertes y recuperados, todos, para un único día. Además, su documentación es muy escueta y requiere un registro previo por lo que, para conocer la marcha del Covid-19 en un país durante 7 días, serían necesarias 7 peticiones de autentificación más otras 7 de petición de datos.

|  |  |  |
| --- | --- | --- |
| **Fortalezas API Propia** | **Debilidades API-nCov2019** | **Fortalezas API-nCov2019** |
| Información extensa con diversas operaciones sobre la misma | Pocas operaciones | Sencillez |
| Una única petición | Necesidad de múltiples peticiones | Exactitud |

4. Recursos

Se pueden clasificar en dos tipos: recursos humanos y recursos materiales. Dentro de los recursos humanos, dada la magnitud del proyecto, se habrá de contar con diversos perfiles: analista de aplicaciones, arquitecto de aplicaciones, desarrollador de base de datos, desarrollador de aplicaciones y, un gestor de proyectos.

Para realizar la aplicación son necesarios los siguientes recursos materiales:

* Cinco ordenadores personales, uno por participante.
* Conexión a internet.
* Lugar de trabajo, escritorio, silla, material de oficina...

Para cada ordenador personal será necesario un sistema operativo. La filosofía se basa en el empleo de programas de uso gratuito y licencia GPL u OSI. Ubuntu será el sistema operativo a emplear. También es necesaria una base de datos. Postgresql ha sido la elegida por sencillez, compatibilidad y uso gratuito. El marco de desarrollo será .Net Core, la versión gratuita de código abierto de Microsoft. GitHub y su paquete gratuito de herramientas, desde gestor de código hasta gestor kanban de proyectos informáticos es también imprescindible para la puesta en común del trabajo conjunto. Queda patente que el uso de licencias no conllevará gasto alguno.

Como recursos materiales adicionales para el despliegue y prueba de la aplicación se ha optado por la adquisición y puesta en marcha de un servidor reducido, para lo que será necesario adquirir una RaspBerry Pi, cuyo precio, contando con todo lo necesario para ponerla en funcionamiento ronda los 119 euros. Además, generará un consumo eléctrico de unos 35 euros anuales, aproximadamente.

## 4.1 Coste recursos humanos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Actividad** | **Responsable** | **Coste por hora** | **Coste total** | **Coste acumulado** |
| Análisis | Analista | 15 euros | 240 euros | 240 euros |
| Diseño | Arquitecto | 20 euros | 320 euros | 560 euros |
| Implementación base de datos | Desarrollador base de datos | 15 euros | 480 euros | 1040 euros |
| Implementación API | Desarrollador | 15 euros | 840 euros | 1880 euros |
| Pruebas de conjunto | Desarrollador base de datos y desarrollador | 30 euros | 240 euros | 2120 euros |
| Muestra | Gestor de proyectos | 20 euros | 320 euros | 2440 euros |
| Subsanación de posibles errores | Desarrollador base de datos y desarrollador | 30 euros | 960 euros | 3400 euros |
| Supervisión proyecto | Gestor proyectos | 20 euros | 3200 euros | 6600 euros |
|  |  |  | **Total:** | **6600 euros** |

## 4.2 Coste en recursos materiales

|  |  |  |  |
| --- | --- | --- | --- |
| **Recurso** | **Coste** | **Mantenimiento** | **Coste Acumulado** |
| Ordenador personal \* 5 | 800 euros \* 5 | 0 euros | 4000 euros |
| Lugar trabajo \* 5 | 300 euros \* 5 | 400 euros mes | 5900 euros |
| Material de oficina | 200 euros | 20 euros | 6120 euros |
| Conexión a internet | 45 euros | 0 | 6165 euros |
| Rasperry Pi | 120 euros | 35 euros | 6320 euros |
|  |  | **Total:** | **6320 euros** |

## 4.3. Coste total del proyecto

|  |  |
| --- | --- |
| **Tipo de coste** | **Coste** |
| Recursos humanos | 6600 euros |
| Recursos materiales | 6320 euros |
| **Total** | **12920 euros** |

# 

5. Análisis de riesgos

## 5.1 Preámbulo

La principal fuente de riesgos a los que se enfrenta el proyecto provienen de la posible falta de conocimientos en alguna de las áreas específicas a tratar. La especialización se ha convertido en algo muy común y eso tiene sus desventajas. No es posible que todos sepan hacerlo todo.

Siguiendo este razonamiento, y puesto que la aplicación requiere de conocimientos que abarcan desde el propio diseño y análisis hasta la fase final de despliegue y mantenimiento de la misma, existe una probabilidad muy alta de que en determinada fase del desarrollo del proyecto sea estrictamente necesario que quien lo desarrolla tenga que formarse en alguna materia. Esto implica una consecuencia, fundamentalmente: un mayor consumo de tiempo.

Pero no es el único, ya que un mal diseño, propio de un principiante puede desembocar en la necesidad de rehacer nuevamente la aplicación, con, de nuevo, el mayor consumo de tiempo.

Así pues, se pueden resumir los riesgos en la necesidad de expandir más o menos el tiempo empleado para cada tarea particular. Sin embargo, y puesto que se habla de la elaboración de un sistema que requiere autentificación y, por tanto, el almacenamiento de los datos de los usuarios la seguridad es clave en este aspecto.

## 5.2 Análisis

|  |  |  |
| --- | --- | --- |
| **Riesgo** | **Probabilidad** | **Grado de importancia** |
| Falta de conocimientos en un momento dado | Muy alta | Muy elevado |
| Necesidad de rehacer el diseño | Media | Muy elevado |
| Falta de seguridad en el tratamiento de los datos | Baja | Muy elevado |
| Mala gestión temporal del proyecto | Media | Elevado |
| Falta de comunicación | Media | Elevado |
| Fallo de los sistemas físicos | Baja | Medio |

6. Plan de contingencia

## 6.1 Preámbulo

Para afrontar los riesgos expuestos en el apartado anterior y derivados de la posible tardanza en la finalización, se han propuesto dos formas de abordarlos: la primera es el empleo de tecnologías ya conocidas y manejadas habitualmente por el desarrollador, lo que minimizará, tanto como sea posible, la necesidad de formarse durante la marcha. Y la segunda, consiste en extender la fase de diseño y análisis tanto como sea posible, ya que, cuantos más problemas se afronten antes de realizar el trabajo de desarrollo, menor necesidad de cambios de última hora serán necesarios.

Para garantizar la seguridad en el manejo de los datos de los usuarios, especialmente en lo que a su dirección de correo electrónico se refiere como a su contraseña, únicamente se recibirán peticiones con este contenido encriptado. El proceso será simple y se corresponderá con la encriptación RSA de doble clave. Cada vez que un usuario sea registrado será proveído de una clave pública con la que encriptar sus comunicaciones. La clave privada se almacenará en un servidor seguro y, será única para cada usuario. El registro de usuario se realizará mediante comunicación encriptada en Base64.

## 6.2 Plan de contingencia

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Riesgo** | **Causa** | **Probabilidad** | **Acción Preventiva** | **Acción Correctiva** | **Medios de aviso** |
| Falta de conocimientos en un momento dado | Especialización laboral | Muy alta | Uso de tecnologías conocidas | Formación | El desarrollador ha de informar |
| Necesidad de rehacer el diseño | Mal análisis. Diseño apresurado. Inexperiencia. | Media | Discusión en fase de análisis.  Diseño sencillo | Refactorizar | El desarrollador ha de avisar. El Gestor de proyectos ha de comprobar que los tiempos marchan adecuadamente. |
| Falta de seguridad en el tratamiento de los datos | Desconocimiento. Inexperiencia | Media | Formación | Refactorizar | El Gestor de proyectos ha de revisar cuidadosamente cómo está implementado el diseño. |
| Mala gestión temporal del proyecto | Inexperiencia.  Personalidad.  Falta de organización. | Media | Experiencia. Organización  Diligencia. | Pedir más tiempo para la finalización | Límites temporales que no se cumplen |
| Falta de comunicación | Introversión.  Desentendimiento | Media | Buen ambiente | Comunicación | Partes en desacuerdo. El gestor de proyectos debe supervisarlo. |
| Fallo de los sistemas físicos | Mala calidad de los equipos.  Imprevisto | Baja | Equipos eficientes y de calidad. | Reposición de la maquinaria estropeada o con falla. | Email. Teléfono. |

7. Análisis y Especificación

## 7.1 Usuarios

Dentro de la aplicación únicamente puede ser registrado un único tipo de usuario: **usuario registrado**. Será el único rol que se contemple como demandante de los servicios funcionales de la misma, sin embargo, es necesario mencionar que la base de datos tendrá un administrador de la misma. También la aplicación dispondrá de un rol relativo al desarrollo de nuevas funcionalidades y mantenimiento de las anteriores.

## 7.2 Entorno Operativo

Para que el desarrollo y la implementación de la aplicación sea independiente del sistema operativo escogido tanto por quien la desarrolle como por quien la mantenga y, asegurar que en un futuro los medios empleados no derivarán en coste alguno por convertirse en privativos, se han escogido dos tecnologías con licencia de código abierto y gratuito.

Para almacenar los datos el motor de Postgresql, por rapidez, sencillez y gratuidad. Mientras que, para el desarrollo de la aplicación y su ejecución .Net Core, la versión de código abierto del entorno de desarrollo de Microsoft.

Se espera que sea desplegada en un ordenador con cualquier versión de Linux pero, en caso de que se decida arrancar en uno con Windows o MacOS debería funcionar sin necesidad alguna de cambio. No se tiene en cuenta el tipo de hardware o composición física del servidor en el que se despliegue pues se ha desarrollado con sufiente nivel de abstracción como para que no sea necesaria consideración alguna acerca de él salvo, quizá, que la computadora posea menos de 10 años.

## 7.3 Requerimientos funcionales

|  |  |
| --- | --- |
| **0001** | *Servir datos por país acerca de la evolución de la pandemia de Covid-19* |
| **1.0** | *1.0 - Junio 2020* |
| **Dependencias** | * *Base de Datos Unión Europea* * *Base de Datos del Gobierno Español* * *Base de Datos Relacional de la Aplicación* * *.Net Core 3.0 y lenguaje de programación C#* * *PostgreSql* |
| **Precondición** | *Poseer los datos ordenados y almacenados en base de datos propia. Petición mediante servicio API Rest de la obtención de los datos demandados. En la petición se deberá de poder identificar al usuario del sistema. El usuario ha de estar registrado en el sistema.* |
| **Descripción** | *Tras recibir una petición Post el sistema deberá identificar al usuario y, en caso positivo servir los datos demandados tras una consulta a la base de datos propia.* |
| **Postcondición** | *Ningún dato habrá quedado modificado, estando el sistema, en todo momento disponible para manejar nuevas peticiones.* |
| **Importancia** | *Máxima* |
| **Prioridad** | *Alta* |
| **Estado** | *Fase final del desarrollo.* |

|  |  |
| --- | --- |
| **0002** | *Servir datos de uno o varios paises dentro de un rango de fechas especificado acerca de la evolución de la pandemia de Covid-19* |
| **1.0** | *1.0 - Junio 2020* |
| **Dependencias** | * *Base de Datos Unión Europea* * *Base de Datos del Gobierno Español* * *Base de Datos Relacional de la Aplicación* * *.Net Core 3.0 y lenguaje de programación C#* * *PostgreSql* |
| **Precondición** | *Poseer los datos ordenados y almacenados en base de datos propia. Petición mediante servicio API Rest de la obtención de los datos demandados. En la petición se deberá de poder identificar al usuario del sistema. El usuario ha de estar registrado en el sistema.* |
| **Descripción** | *Tras recibir una petición Post el sistema deberá identificar al usuario y, en caso positivo servir los datos demandados tras una consulta a la base de datos propia. La demanda de estos datos puede estar segmentada para un país o varios dentro de un rango de fechas concreto.* |
| **Postcondición** | *Ningún dato habrá quedado modificado, estando el sistema, en todo momento disponible para manejar nuevas peticiones.* |
| **Importancia** | *Máxima* |
| **Prioridad** | *Alta* |
| **Estado** | *Fase final del desarrollo.* |

|  |  |
| --- | --- |
| **0003** | *Servir datos de uno o varios paises dentro de un rango de fechas especificado y con una operación sobre los mismos realizada acerca de la evolución de la pandemia de Covid-19* |
| **1.0** | *1.0 - Junio 2020* |
| **Dependencias** | * *Base de Datos Unión Europea* * *Base de Datos del Gobierno Español* * *Base de Datos Relacional de la Aplicación* * *.Net Core 3.0 y lenguaje de programación C#* * *PostgreSql* |
| **Precondición** | *Poseer los datos ordenados y almacenados en base de datos propia. Petición mediante servicio API Rest de la obtención de los datos demandados. En la petición se deberá de poder identificar al usuario del sistema. El usuario ha de estar registrado en el sistema.* |
| **Descripción** | *Tras recibir una petición Post el sistema deberá identificar al usuario y, en caso positivo servir los datos demandados tras una consulta a la base de datos propia. La demanda de estos datos puede estar segmentada para un país o varios dentro de un rango de fechas concreto. En la petición recibida por el sistema puede especificarse una de las siguientes operaciones y la aplicación debe de ser capaz de gestionarla: media y mediana.* |
| **Postcondición** | *Ningún dato habrá quedado modificado, estando el sistema, en todo momento disponible para manejar nuevas peticiones.* |
| **Importancia** | *Media* |
| **Prioridad** | *Media* |
| **Estado** | *Fase final del desarrollo.* |

## 7.4 Requerimientos no funcionales

|  |  |
| --- | --- |
| **0001** | *Eficiencia en el servicio de los datos* |
| **1.0** | *1.0 - Junio 2020* |
| **Dependencias** | * *Base de Datos Relacional de la Aplicación* * *.Net Core 3.0 y lenguaje de programación C#* * *PostgreSql* * *Sistema Operativo* * *Técnica de programación* |
| **Descripción** | *El sistema ha de ser capaz de identificar al usuario, obtener los datos demandados y generar una respuesta en menos, siempre, de dos segundos máximo. Como meta se espera una respuesta rápida e inferior a 500 ms.* |
| **Importancia** | *Alta* |
| **Prioridad** | *Media* |
| **Estado** | *Fase final del desarrollo.* |

|  |  |
| --- | --- |
| **0002** | *Capacidad para gestionar peticiones simultáneas* |
| **1.0** | *1.0 - Junio 2020* |
| **Dependencias** | * *Base de Datos Relacional de la Aplicación* * *.Net Core 3.0 y lenguaje de programación C#* * *PostgreSql* * *Sistema Operativo* * *Técnica de programación* |
| **Descripción** | *No debe de existir error alguno en caso de que n peticiones se realicen al mismo tiempo. Como mínimo debe de ser capaz de gestionar 50 peticiones simultáneas.* |
| **Importancia** | *Alta* |
| **Prioridad** | *Media* |
| **Estado** | *Fase final del desarrollo.* |

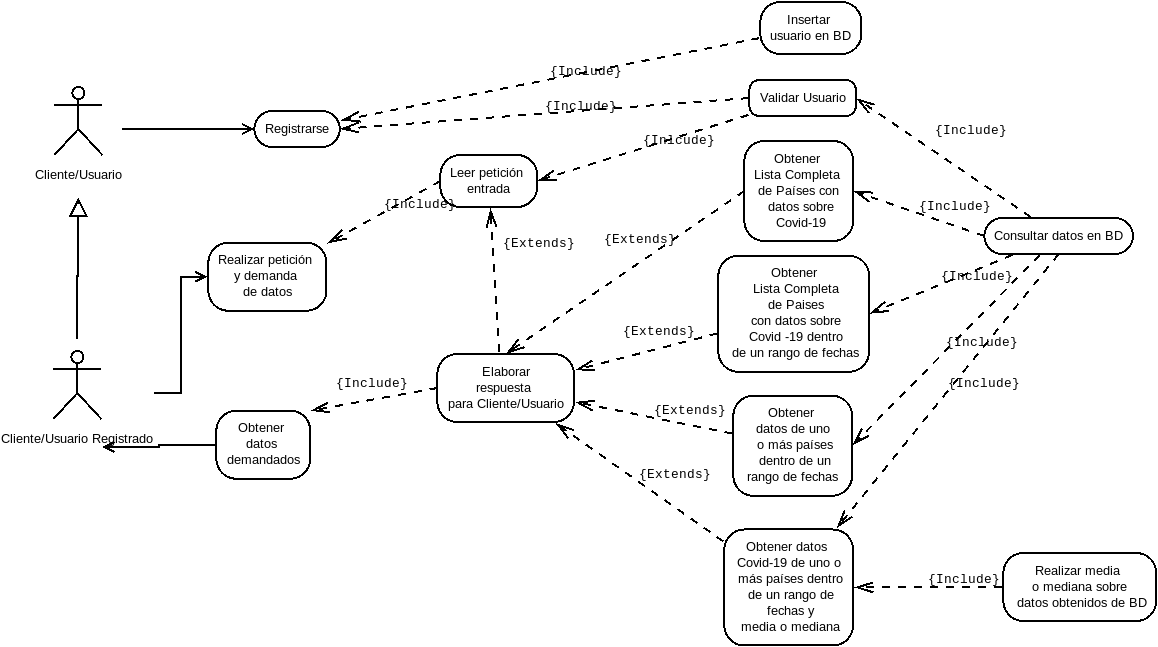
|  |  |
| --- | --- |
| **0003** | *Seguridad en el almacenamiento de los datos* |
| **1.0** | *1.0 - Junio 2020* |
| **Dependencias** | * *Base de Datos Relacional de la Aplicación* * *.Net Core 3.0 y lenguaje de programación C#* * *PostgreSql* * *Sistema Operativo* * *Técnica de programación* |
| **Descripción** | *Los datos sensibles relacionados con el almacenaje del usuario y su contraseña, deben de ser seguros y permanecer aislados de la aplicación tanto como sea posible.* |
| **Importancia** | *Máxima* |
| **Prioridad** | *Alta* |
| **Estado** | *Fase final del desarrollo.* |

|  |  |
| --- | --- |
| **0004** | *Facilidad en el uso* |
| **1.0** | *1.0 - Junio 2020* |
| **Dependencias** | * *Base de Datos Relacional de la Aplicación* * *.Net Core 3.0 y lenguaje de programación C#* * *PostgreSql* * *Sistema Operativo* * *Técnica de programación* |
| **Descripción** | *La aplicación debe de permitir una facilidad de empleo garantizando que su uso sea lo más sencillo posible. De ahí que únicamente admita un tipo de petición que centralice todas las posibilidades de demanda de datos.* |
| **Importancia** | *Máxima* |
| **Prioridad** | *Alta* |
| **Estado** | *Fase final del desarrollo.* |

|  |  |
| --- | --- |
| **0005** | *Disponible siempre* |
| **1.0** | *1.0 - Junio 2020* |
| **Dependencias** | * *Base de Datos Relacional de la Aplicación* * *.Net Core 3.0 y lenguaje de programación C#* * *PostgreSql* * *Sistema Operativo* * *Técnica de programación* |
| **Descripción** | *Debe de ser posible usar la aplicación a cualquier hora del día cualquier día del año. En caso de falla ésta no debe detenerla.* |
| **Importancia** | *Máxima* |
| **Prioridad** | *Alta* |
| **Estado** | *Fase final del desarrollo.* |

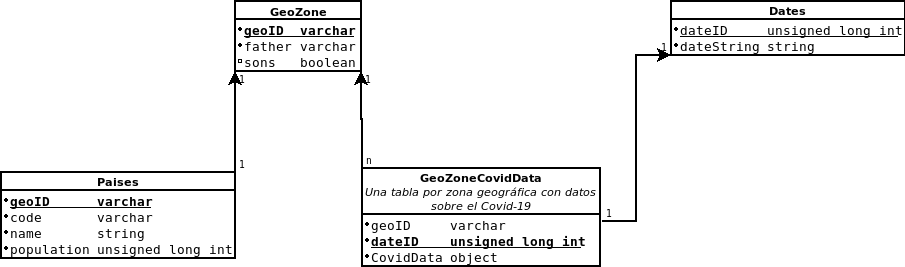
|  |  |
| --- | --- |
| **0006** | *Independencia del SO* |
| **1.0** | *1.0 - Junio 2020* |
| **Dependencias** | * *Base de Datos Relacional de la Aplicación* * *.Net Core 3.0 y lenguaje de programación C#* * *PostgreSql* * *Sistema Operativo* * *Técnica de programación* |
| **Descripción** | *La aplicación debe funcionar en cualquier sistema operativo.* |
| **Importancia** | *Máxima* |
| **Prioridad** | *Alta* |
| **Estado** | *Fase final del desarrollo.* |

## 7.5 Casos de uso



## 7.6 Modelo Entidad Relacional

### 7.6.1 Modelo E/R Datos Covid



### 

### 7.6.2 Modelo E/R Base de Datos Usuarios



8. Diseño

## 8.1 Modelo de datos de entrada

Las peticiones de entrada se realizarán en formato JSON. Es sencillo, simple y relativamente parecido al modelado de objetos. Será el siguiente:

[

{

"user":

{

"id": "mail@example.net",

"pass": "passExample"

}

,

"covid\_data":

{

"countries":

[

"countryA",

"countryB",

"countryC"

],

"dates":

{

"start\_date": "dd/MM/yyyy",

"end\_date": "dd/MM/yyyy"

},

"operation\_type": "operationName"

]

## 8.2 Modelo de datos de salida

{

"father" : null,

"son\_list" : null,

"geoID" : "AD",

"code" : "AND",

"name" : "Andorra",

"population" : 77006,

"dataList" : [

{

"date" : {

"id" : 1,

"date" : "03/03/2020",

"dateSeparator" : "/",

"dateFormat" : "dd/MM/yyyy"

},

"cases" : 1,

"deaths" : 0,

"cured" : 0

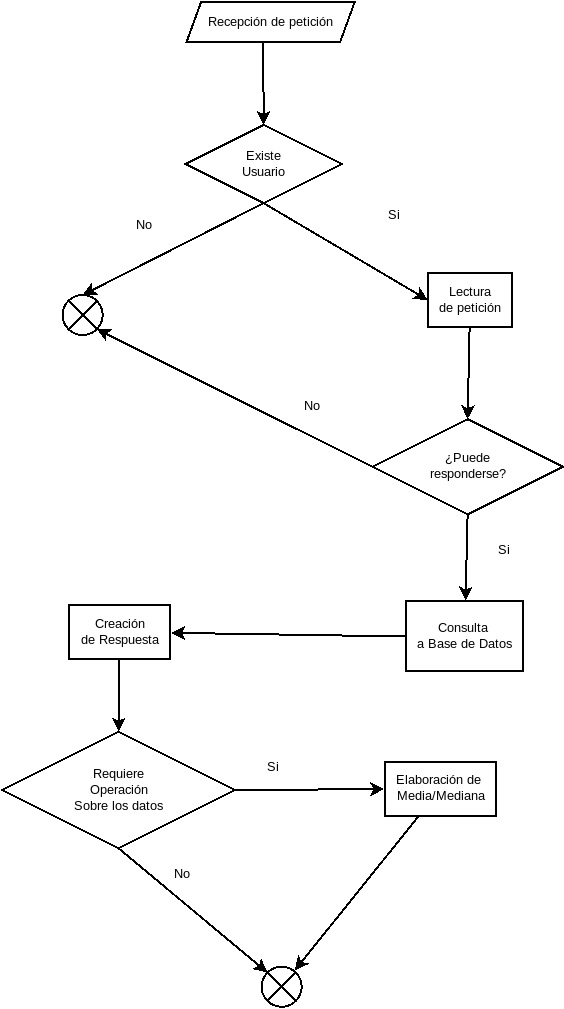
},

….

]

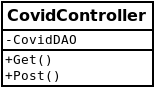
}

## 8.3 Diagrama de flujo

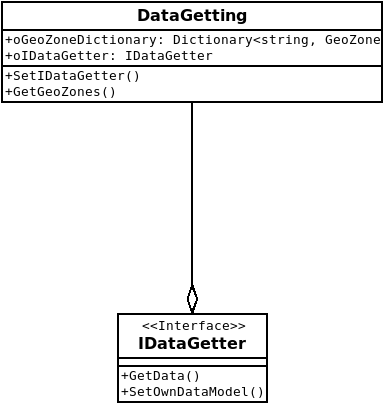


## 8.4 Diagramas de clases

### 8.4.1 Diagrama de clases Controlador

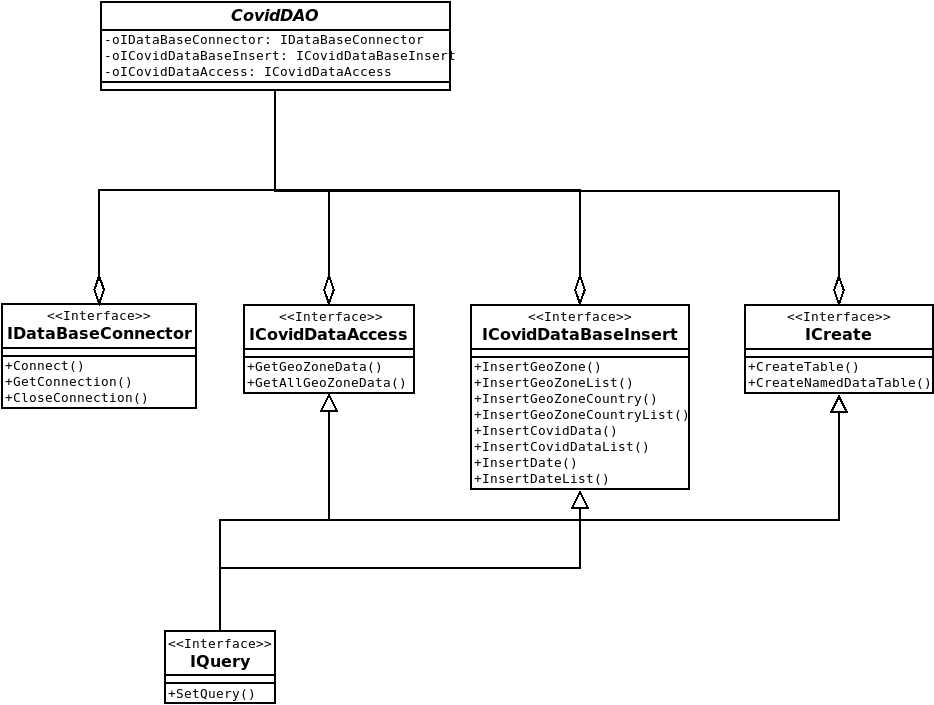


### 8.4.2 Diagrama de clases Proceso



### 

### 8.4.3 Diagrama de clases DAO (Data Access Object)



### 

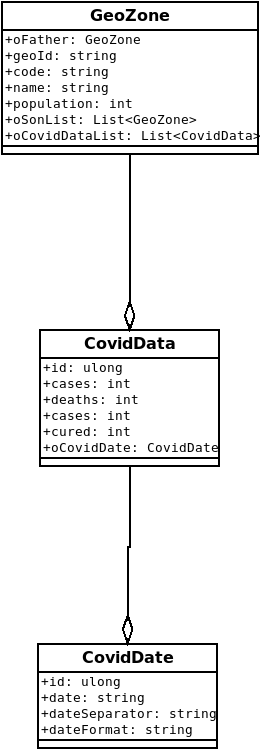
### 

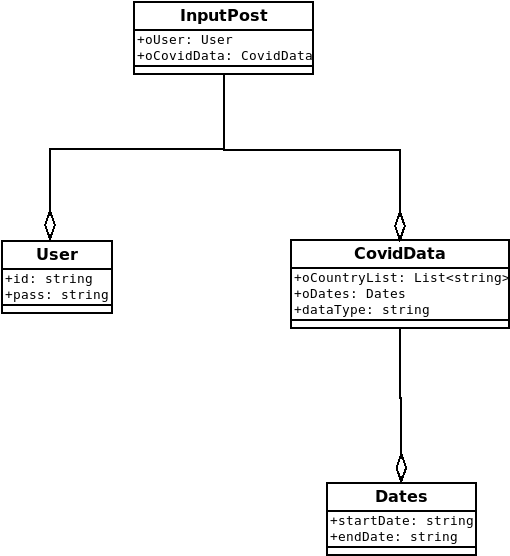
### 

### 

### 

### 8.4.4 Diagramas de clases Modelo





9. Fase de Pruebas

La fase de pruebas tiene como objetivo garantizar la calidad y el adecuado funcionamiento de la aplicación antes de ser puesta a disposición de los usuarios. Se seguirá una estrategia simple de desarrollo y prueba inmediata. Es decir, cada fragmento que sea desarrollado de la aplicación será probado una vez finalizado el mismo. Tras haber completado el desarrollo se realizarán también pruebas que comprueben el funcionamiento en conjunto del programa informático.

Se espera que al término de esta fase la aplicación se encuentre libre de errores y fallas y pueda usarse con total confianza.

## 9.1 Características a ser probadas

Correspondiéndose con los requisitos funcionales las características a ser probadas son:

* Recibir los datos de la evolución de la pandemia del país que se ha solicitado.
* Recibir los datos pertenecientes a varios países dentro de un rango de fechas cuando así se hayan solicitado.
* Recibir los datos acerca de la evolución del Covid pertenecientes al país que se ha demandado, dentro del rango de fechas solicitado y, el resultado de la operación demanda.

## 9.2 Características a no ser probadas

* Eficiencia en el servicio de los datos. Es una característica para la que se necesita una recogida previa y abundante de datos, por lo que se probará una vez esté activa la aplicación.
* Capacidad para gestionar peticiones simultáneas. De nuevo, la mejor prueba es el uso.
* Facilidad en el uso. Se consultará a los clientes una vez hayan empleado un tiempo razonable en usarla.

## 9.3 Casos de Prueba

|  |  |  |
| --- | --- | --- |
| **DevuelvePaisSolicitado** | **001** |  |
| **Prueba de despliegue** | **Sí** |
| **Descripción:**  Se realizarán diez peticiones para diez países distintos y se comprobará que en cada una de las respuestas los datos pertenezcan al país solicitado. | | |
| **Prerrequisitos**:  Tener en pleno funcionamiento la aplicación. | | |
| **Pasos:** | **Datos**: | |
| Crear Petición | Paises disponibles | |
| Enviar Petición | Usuario generado | |
| Recibir Respuesta | Modelo de entrada de datos | |
| Comprobar Respuesta | Modelo de salida de datos | |
| Repetir Proceso |  | |

|  |  |  |
| --- | --- | --- |
| **DevuelvePaisesPorFechaSolicitados** | **002** |  |
| **Prueba de despliegue** | Si |
| **Descripción:**  Se realizarán diez peticiones para 100 países distintos de diez en diez y para diferentes rangos de fechas. Posteriormente se comprobará que en cada una de las respuestas los datos pertenezcan a los países solicitados y estén dentro del rango de fechas especificado. | | |
| **Prerrequisitos**:  Tener en pleno funcionamiento la aplicación. | | |
| **Pasos:** | **Datos**: | |
| Crear Petición | Paises disponibles | |
| Enviar Petición | Usuario generado | |
| Recibir Respuesta | Modelo de entrada de datos | |
| Comprobar Respuesta | Modelo de salida de datos | |
| Repetir Proceso |  | |

|  |  |  |
| --- | --- | --- |
| **DevuelvePaisesPorFechaConOperacionSolicitados** | **003** |  |
| **Prueba de despliegue** | Si |
| **Descripción:**  Se realizarán diez peticiones para 100 países distintos de diez en diez y para diferentes rangos de fechas. Para cada petición se determinará una de las dos operaciones solicitables. Posteriormente se comprobará que en cada una de las respuestas los datos pertenezcan a los países solicitados y estén dentro del rango de fechas especificado. También se contrastarán los resultados de la media o la mediana según proceda. | | |
| **Prerrequisitos**:  Tener en pleno funcionamiento la aplicación. | | |
| **Pasos:** | **Datos**: | |
| Crear Petición | Paises disponibles | |
| Enviar Petición | Usuario generado | |
| Recibir Respuesta | Modelo de entrada de datos | |
| Comprobar Respuesta | Modelo de salida de datos | |
| Repetir Proceso | Tipos de operación | |

## 9.4 Necesidades ambientales

Para realizar las pruebas únicamente serán necesarias:

* Una computadora con acceso a internet.
* La instalación de .Net Core y ASP.Net .
* Un programa de envío de peticiones. Postman en este caso.
* Un IDE de desarrollo.
* Una Base de datos configurada Postgresql.
* Un Sistema operativo.

10. Bibliografía

Es posible realizar una clasificación de las fuentes de información atendiendo al tipo de datos que ofrecen.

## 10.1 Datos oficiales acerca de la evolución del Covid-19:

**Organismos oficiales de obtención de big data:**

### Datos procedentes de organismos de la Unión Europea:

https://data.europa.eu/

### Datos procedentes del gobierno español:

https://datos.gob.es/

## 10.2 Datos acerca de los marcos técnicos a emplear:

### Documentación oficial de .Net Core:

https://docs.microsoft.com/es-es/dotnet/core/

### Documentación oficial de MongoDB, Postgre y MariaDB:

https://www.mongodb.com/es

<https://www.postgresql.org/>

https://mariadb.org/