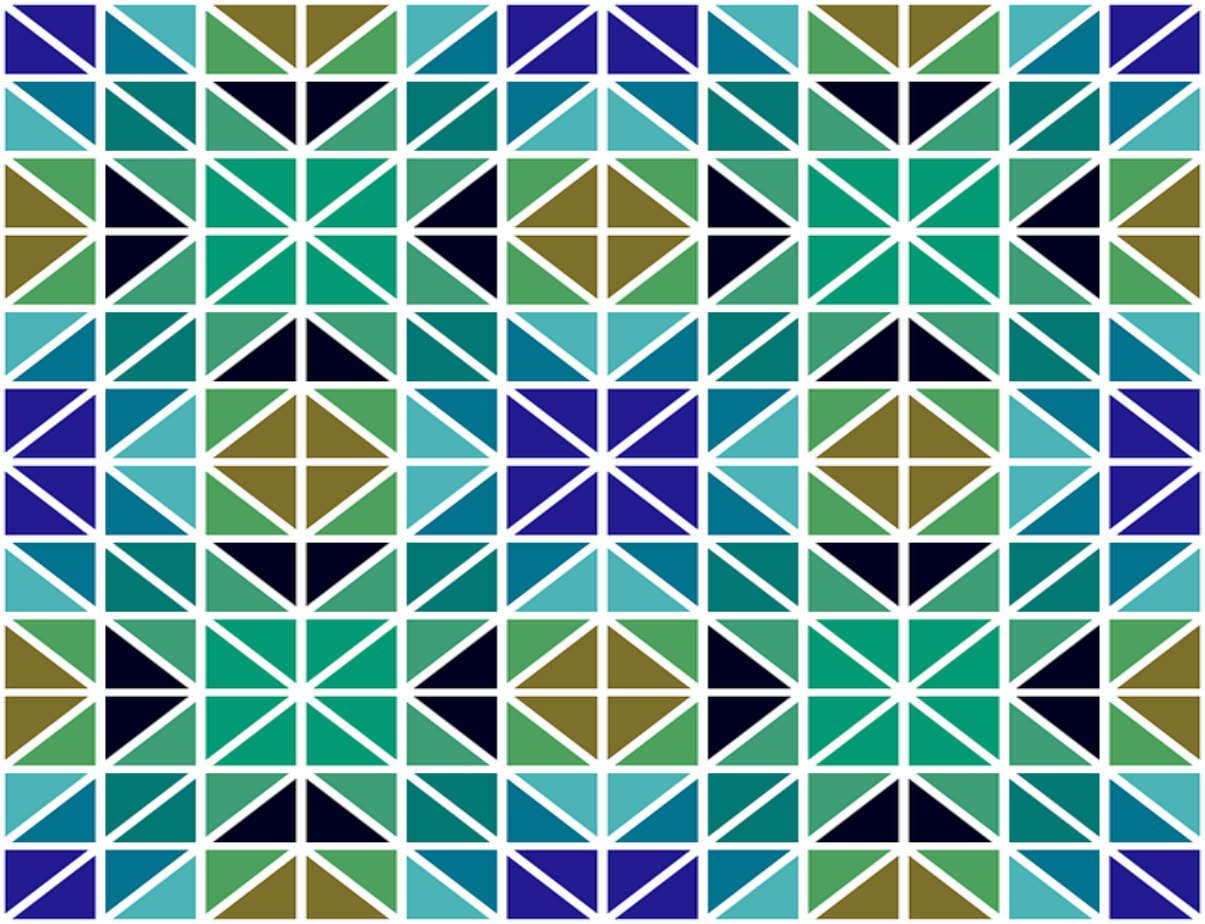
CIFP Pau Casesnoves



API Covid - 19 y API Seguridad

Autor: Álvaro Maleno Alférez

Grado Superior Desarrollo de Aplicaciones Multiplataforma CIFP Pau Casesnoves Inca. Illes Balears.

Tutorización:

Joan Cesari Bohigas

Miquel Óscar García Jódar

Índice

1. Datos personales del centro y de la empresa

1.1 Centro en el que se desarrolla

- CIFP Pau Casesnoves. Inca. CP 07300. Illes Balears.

1.2 Tutores

- Joan Cesari Bohigas
- Miquel Óscar García Jódar

1.3 Alumno

- Álvaro Maleno Alférez

1.4 Empresa

- AxisData S.L.
- Puesto del alumno: Desarrollador Junior

2. 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”. Este proyecto consiste en una aplicación que permita el registro de usuarios y la consulta, por parte de los mismos, de datos acerca de la pandemia de Covid -19.

Los datos consultables serán los relativos al número de contagiados por día y país. También se incluirá el número de personas recuperadas y la sumatoria total. La seguridad conlleva un proceso de encriptación de los datos del usuario. Cada uno de estos dispondrá de su pareja de claves pública y privada. Estas claves, siendo almacenadas en la base de datos de la aplicación encriptadas, serán, cada cierto número de usos almacenadas con una nueva encriptación. Nunca variarán las claves de usuario.

Se desarrollará dentro del contexto de empresa orientada a la venta de productos informáticos. Cada producto se identifica con un microservicio por lo que la arquitectura con dicho nombre será la empleada. Para cualquier empresa, una aplicación como ésta sirve de escaparate para futuros clientes que podrán comprobar su fiabilidad y forma de proceder.

2.1 Descripción

Los datos consultables serán los relativos al número de contagiados por día y país. También se incluirá el número de personas recuperadas y la sumatoria total. La seguridad conlleva un proceso de encriptación de los datos del usuario. Cada uno de estos dispondrá de su pareja de claves pública y privada. Estas claves, siendo almacenadas en la base de datos de la aplicación encriptadas, serán, cada cierto número de usos almacenadas con una nueva encriptación. Nunca variarán las claves de usuario.

Se trata de un servicio API HTTP que requiere cuenta de usuario para su acceso. Ofrecerá una serie de datos estadísticos actualizados y bajo demanda. Además, necesitaría de un servicio añadido que permitiese la gestión segura de las cuentas de usuario. De ahí que se desarrolle también una API HTTP que, encriptando en base de datos la información de dicho usuario garantice su seguro almacenamiento realizando cambios periódicos en la encriptación de la

misma. Y para que el usuario no deba exponer sus datos en ningún momento primero recibirá una clave pública con la que encriptarlos para darse de alta y una segunda clave pública personal para que pueda encriptar los datos de autentificación en futuras comunicaciones.

Con el objetivo de disminuir el número de identificaciones un servicio de tokenización se implementará, de manera que el usuario únicamente tenga que autenticarse una vez cada 24 horas. Ésta tokenización será puesta en práctica en un tercer servicio API HTTP que servirá de interfaz para el acceso a los dos mencionados anteriormente, de manera que estos queden menos expuestos al público y puedan ser utilizados por otras aplicaciones al mismo momento.

2.2 Objetivos

2.2.1 Objetivos Primarios:

- Desplegar tres servicios API HTTP.
- 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.
- Garantizar la seguridad en el tratamiento de los datos de los usuarios.
- Posibilidad de emplear las distintas APIs en otras aplicaciones.
- Permitir a los usuarios acceder a los datos de forma rápida y segura.
- Generar un API sencillo y manejable para el usuario.
- Presentar los datos de forma simple y legible.

2.2.2 Objetivos Técnicos:

- Acceder a la información y a las operaciones que puedan derivarse de sus datos.
- Consultar los datos relativos a la pandemia.
- Ofrecer consultas dentro de unas fechas determinadas.
- Consultar para un lugar geográfico concreto.

- Consultar para varios lugares geográficos.
- Consultar toda la información para todos los lugares geográficos disponibles.
- Consultar los lugares geográficos disponibles.
- Consultar las fechas con datos disponibles.
- No exponer nunca la información de usuario.
- Generar un sistema de encriptación dinámico para el almacenamiento de la información sensible que pueda ser usado en otras aplicaciones.
- Asignar un par de claves pública y privada a cada usuario.
- Que las claves almacenadas cambien su encriptación cada N usos.
- Disminuir el número de autenticaciones necesarios mediante el uso de un token con 24 horas de durabilidad.
- Arquitectura microservicios.
- Dockerización.

2.3 Justificación

En el ámbito empresarial presentar un producto propio a los posibles clientes es una necesidad primordial a la hora de mostrar sus capacidades. Además, ofrecer un servicio gratuito y abierto a todos los públicos genera un impacto positivo en la sociedad y, una consiguiente función social más amplia de la empresa.

Para las personas es importante poder acceder a la información cuando la necesita. Esto quiere decir que debe de estar siempre disponible y de ser gratuita. Conocer de primera mano si es una buena idea o no viajar a otro país o el entorno de personas con las que se posee alguna relación resulta de vital importancia. Es también una necesidad que poseen las empresas que basan sus estrategias comerciales en los cambios producidos en su entorno.

En mitad de una pandemia (Covid -19) como la actual resulta indispensable la posibilidad de consultar los datos que le son relativos de manera simple, sencilla y personalizada. Los organismos oficiales ofrecen únicamente los datos en bruto y éstos nunca son filtrados u ordenados. Los medios de comunicación tampoco realizan esta labor más allá de su inclusión en una noticia o una gráfica visualizable desde su página web. Para cubrir este hueco se desarrolla este proyecto.

La seguridad en el tratamiento de los datos es también importante. Dar servicio de autentificación a diversas páginas web o APIs de manera general y garantizando que los datos de usuario quedan a buen recaudo justifica la creación de un sistema de
seguridad que los almacene con una encriptación propia y variable a lo largo del tiempo.

2.4 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. Cualquier página web que requiera de autentificación mediante usuario y contraseña y, en definitiva, cualquier servicio que requiera del almacenamiento de unos datos sensibles puede beneficiarse del desarrollo en la rama de seguridad de este proyecto.

Puede suplir además varias necesidades. Para los gestores de páginas web les puede resultar provechoso ahorrar tiempo empleando la identificación ofrecida por este proyecto ya que no necesitarán desarrollarlo por sí mismos. Además, podrán ofrecer los datos a sus visitantes segmentados por el país de los mismos. A los ciudadanos les favorecería su implementación ya que tendrían información oficial y consistente en sus portales web favoritos.

2.5 Ámbito del proyecto

2.5.1 Metodología del proyecto

En el terreno de las empresas tecnológicas desde tiempo atrás se han venido implementando 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.

Se seguirá el desarrollo por iteración. Es decir, se comenzará por desarrollar una parte pequeña de la aplicación y, si cumple con lo requerido se pasará a la siguiente fase de desarrollo, haciendo crecer la aplicación. En caso contrario, se realizaría una segunda iteración sobre la parte desarrollada.

Se complementará con el desarrollo en cascada, comenzando por la fase de análisis, diseño, implementación, pruebas y puesta en marcha. El trabajo se dividirá en tareas pequeñas - método Kanban - de manera fácilmente asumible por el equipo de desarrollo. Se mantendrán reuniones semanales para conversar acerca de las tareas. Las pruebas del programa la realizarán los propios desarrolladores.

2.5.2 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. Así, serán empleados los conocimientos aprendidos en el ciclo de Desarrollo de Aplicaciones Multiplataforma:

- Bases de Datos
- Programación multiproceso
- Entornos de desarrollo
- Programación

Se requerirán habilidades de análisis e identificación de problemas. Saber priorizar los requerimientos funcionales, así como saber dividir las tareas y las funciones será una tarea de obligatoria aplicación a lo largo del proyecto. Aplicar las técnicas de diseño aprendidas en las asignaturas cursadas y durante la experiencia laboral será otro punto a tener en consideración. Por último, el desarrollo dentro de diversas tecnologías y lenguajes de programación es, también, un requerimiento de conocimiento básico.

Será necesario poseer familiaridad con:

- El marco de desarrollo de .Net Core.
- El gestor de base de datos PostgreSQL.
- Programación paralela.

- Contenerización de aplicaciones con Docker.
- Diseño y documentación de aplicaciones con Open API.
- Gestión de diagramas UML con Draw.io.
- Gestión del código fuente mediante Git.

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 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.

2.5.3 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.

Con respecto a los datos de usuario, se ha de tener en cuenta la Ley Orgánica de Protección de Datos y Garantía de Derechos Digitales. Se trata de la ley española que aplica el Reglamento General de Protección de Datos Europeo vigente desde mayo de 2018. Su finalidad es proteger la intimidad, integridad y privacidad del individuo regulando el intercambio y la transferencia de datos. Por lo tanto, cualquier dato personal, como el correo electrónico debe de estar a buen recaudo, recayendo esa responsabilidad sobre la empresa o individuo que lo gestiona y almacena. Tampoco puede ser transferido a terceros como medio de intercambio de información ni de ninguna otra manera.

3. 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.

3.1 Diseño

Un diseño comienza siempre con la tarea previa de análisis. Es necesario saber que 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 de software.

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 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, van a ser empleados dos tipos de diagrama UML: diagrama de flujo y diagrama de clases.

Como los requerimientos funcionales habrán sido previamente informados, el diagrama de flujo se ocupará de ponerlos en relación con los pasos que seguirá el algoritmo que compondrá 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 modelo de datos adecuado. Como la base de datos no será demasiado extensa, bastará un único día para desarrollar esta actividad.

3.2 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 las APIs generando, primero, las clases necesarias e implementando posteriormente los procesos que sean requeridos. Se estima que la primera tarea ocupará una semana y cuatro la segunda, pues necesita de mayor algoritmia.

Las pruebas de calidad y funcionalidades se realizarán 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.

3.3 Cumplimiento de tareas

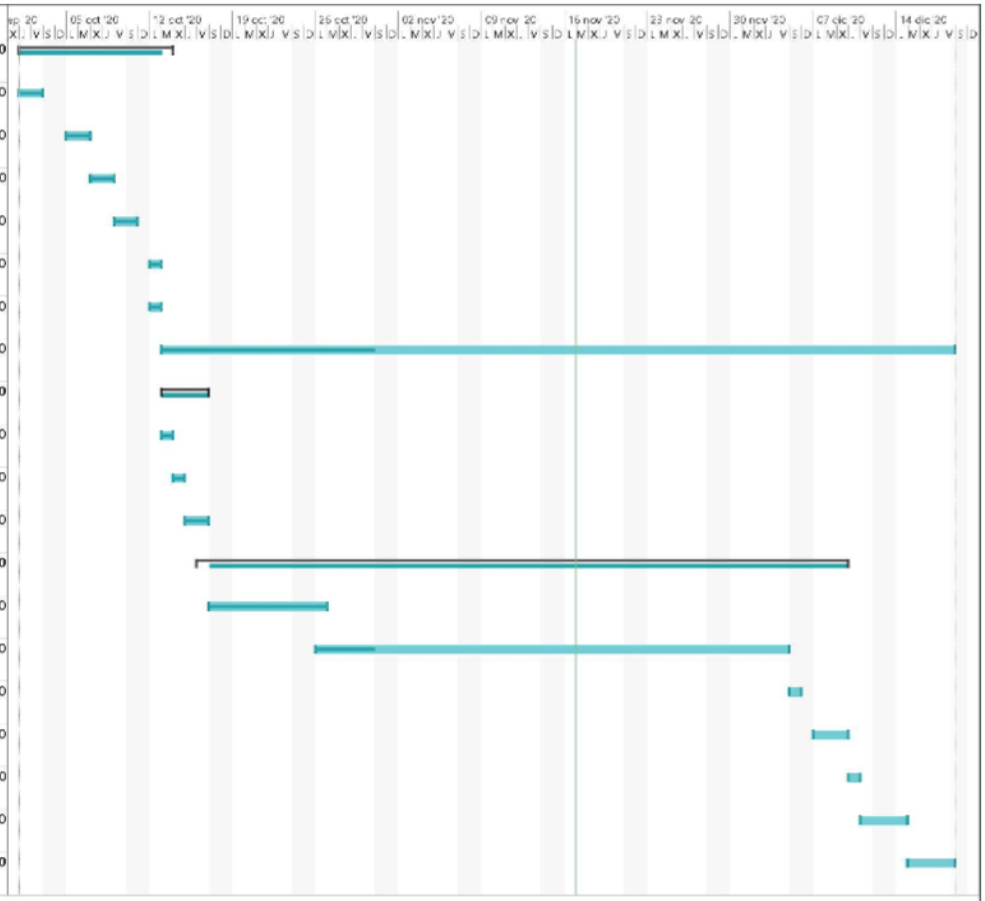
A pesar de que se ha dado margen de maniobra para la subsanación de los errores durante el mismo desarrollo del proyecto los retrasos y el surgimiento de imprevistos es inevitable. Es, entonces, imprescindible el perfil del Gestor de Proyectos.

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.

3.4 Diagrama de Gantt

En la página siguiente puede verse el diagrama de Gantt seguido para la realización del proyecto.

Id	Work Item ID	Title	Duración	Comienzo	Fin
1	1	**DISSEÑO**	**9 días**	**jue 01/10/20**	**mar 13/10/20**
2	100	Análisis de requerimientos	2 días	jue 01/10/20	vie 02/10/20
3	101	Análisis funcional	2 días	lun 05/10/20	mar 06/10/20
4	102	Diagrama de flujo	2 días	mié 07/10/20	jue 08/10/20
5	103	Diagrama de clases	2 días	vie 09/10/20	sáb 10/10/20
6	104	Diseño de Base de datos	1 día	lun 12/10/20	lun 12/10/20
7	105	Diagrama Entidad Relación	1 día	lun 12/10/20	lun 12/10/20
8	43	IMPLEMENTACIÓN	49 días	mar 13/10/20	vie 18/12/20
9	106	**Persistencia**	**4 días**	**mar 13/10/20**	**vie 16/10/20**
10	107	Creación Base de Datos	1 día	mar 13/10/20	mar 13/10/20
11	108	Despliegue de Base de Datos	1 día	mié 14/10/20	mié 14/10/20
12	109	Pruebas Inserción y Lectura Base Datos	2 días	jue 15/10/20	vie 16/10/20
13	110	**API Rest**	**39 días**	**vie 16/10/20**	**mié 09/12/20**
14	111	Generación clases	7 días	sáb 17/10/20	lun 26/10/20
15	112	Implementación de Procesos	30 días	lun 26/10/20	vie 04/12/20
16	113	Prueba de funcionalidades	1 día	sáb 05/12/20	sáb 05/12/20
17	114	Despliegue de la Aplicación	3 días	lun 07/12/20	mié 09/12/20
18	115	**Pruebas de Conjunto**	1 día	jue 10/12/20	jue 10/12/20
19	116	**Presentación y Muestra**	2 días	vie 11/12/20	lun 14/12/20
20	117	**Subsanación de posibles Errores**	**4 días**	**mar 15/12/20**	**vie 18/12/20**



Proyecto: Simple Project Plan
Fecha: mar 17/11/20

Tarea	Resumen del proyecto	Tarea manual	solo el comienzo	Fecha límite			
División	Tarea inactiva	solo duración	solo fin	Progreso			
Hito	Hito inactivo	Informe de resumen manual	Tareas externas	Progreso manual			
Resumen	Resumen inactivo	Resumen manual	Hito externo				

4. 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 proveída 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. La seguridad en la consulta y en el tratamiento de los datos del usuario queda también en el aire.

Surge entonces un nicho de mercado muy preciso en el que la competencia es mínima. 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 consumo de la API de seguridad que la acompaña.

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, al menos no sin tener que lidiar con la interminable generación de urls relativamente largas. 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 total seguridad y eficiencia. Además, estos datos estarán siempre almacenados para futura consulta.

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.

4.1 https://covid-19-apis.postman.com/

Su documentación se encuentra accesible desde https://postman-toolboxes.github.io/covid-19/#featured-collections

Desde ella se puede observar que la cantidad de operaciones permitidas se resume básicamente en dos: obtener el conjunto de datos para uno o varios países 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 únicamente son accesibles a través de días concretos o 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 rango de fecha. El principal problema surge al tener que lidiar con la cantidad de urls a conformar para la obtención de cada uno de los datos. Y no solamente lidiar, sino que el propio modelo de datos de la respuesta es susceptible de ser variable también. Así, para cada tipo de información que ofrecen será necesario el desarrollo de un servicio de petición distinto en base al cual conformar la url que será utilizada para el mismo.

Fortalezas API propia	**Debilidades API Postman**	**Fortalezas API Postman**
Consulta por rango de fechas para un país.	Necesidad de lidiar con el más tedioso trabajo de modificar y manipular distintas urls.	Gran cantidad de información fiable.
Una única petición para obtener toda la información.	Necesidad de realizar múltiples peticiones para obtener un mapa completo.	API muy eficiente.

4.2 API-nCov2019

Accesible desde https://apimarket.nubentos.com/store/apis/info?name=API-nCoV2019&version=2.0.0&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.

5. 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.

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...
- Sistema operativo Ubuntu.
- Base de datos Postgresql.
- Marco de desarrollo .Net Core 3.0 .
- Gestor de código fuente Git con Github.
- Cuenta de Microsoft Azure para despliegue de aplicaciones.
- Cuenta con servicios Amazon AWS para despliegue de base de datos.

El siguiente personal será imprescindible:

- Analista
- Arquitecto de software
- Desarrollador de base de datos
- Desarrollador de aplicaciones .Net
- Gestor de proyectos

6. Análisis y Especificación

6.1 Usuarios

Existen dos tipos de usuario:

1. Todos aquellos desarrolladores o empresas de desarrollo que requieran incorporar cierta seguridad en el trato de los datos del proceso de autentificación de usuario. Direcciones de correo electrónico personales, contraseñas, direcciones, números de teléfono, etc. Todo ello se encuentra automatizado ya en la API de seguridad que contiene este proyecto a modo de bloque. Cualquier dato que sea necesario resulta fácilmente desarrollable con una mínima cantidad de trabajo. Por tanto, todo aquel que requiera implementar un proceso de autentificación de usuario es, a su vez, un posible usuario de esta aplicación.

2. Científicos de datos que requieran de un acceso ágil y veloz a la información acerca de la evolución de la pandemia global, páginas web que deseen mostrar información en tiempo real, cargando solamente los datos que les son imprescindibles lo más rápida y menos pesadamente posible, desarrolladores que quieran ofrecer un producto sencillo a agencias de viajes, cuadernos de bitácora - blogs – o pequeños negocios que precisen o deseen, como servicio adicional y forma de solidarizarse con las víctimas mostrar los datos desde su escaparate web. En definitiva, todo aquel que ofrezca una presencia en línea de sus productos, negocios, artículos, etc.

En cualquier caso el nivel de conocimientos informáticos relativos a la programación ha de situarse entorno a un rango que varía entre principiante y medio. Si bien cualquier principiante puede fácilmente acceder a su uso, aquellos que se encuentren en un nivel algo más avanzado encontrarán también ciertas ventajas y posibles aplicaciones distintas a las que ya posee.

6.2 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 Mac OS 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 suficiente nivel de abstracción como para que no sea necesaria consideración alguna acerca de él salvo, quizá, que la computadora posea menos de 10 años.

6.3 Requerimientos funcionales

6.3.1 De Seguridad

1. Proceso de alta de usuario cifrado. Jamás se recibirán o enviarán datos sensibles sin encriptar.

2. Obtención de clave de cifrado para realizar el registro de un nuevo usuario. Mediante una url única se le ofrecerá al usuario la posibilidad de obtener una clave pública de cifrado perecedera con la cual encriptar sus datos por primera vez.

3. Poseer unas claves pública y privada para uso propias de la aplicación que variarán cada determinado número de usos.

4. Vincular una pareja de claves privada y pública únicas con cada usuario. Cada usuario poseerá, dentro del sistema, sus propias claves de encriptado y desencriptado.

5. Otorgar su correspondiente clave pública a cada usuario. Cada usuario registrado deberá poder conservar una clave pública con la que encriptar cada una de sus autenticaciones.

6. Recibir peticiones encriptadas de autentificación de usuario.

7. Validar la autenticación de un usuario desencriptando la comunicación recibida con su clave privada única.

8. Almacenar las claves pública y privada de usuario encriptadas con una clave propia de la aplicación que será perecedera.

9. Actualizar la encriptación de las claves de usuario cada vez que la aplicación cambia de claves pública y privada con las que funcionar.

10. Cambiar de claves pública y privada propias de la aplicación cada cierto número de usos.

11. Otorgar un token JWT perecedero cada 24 horas. Se trata de disminuir la necesidad de exponer la información sensible de usuario y facilitar el acceso a los datos del sistema.

12. El proceso de generación de tablas de usuario y de aplicación en base de datos deberá estar automatizado.

6.3.2 Organización y obtención de los datos

1. No se podrá acceder a los mismos sin haber sido dado de alta en el sistema y estar autentificado.

2. El modelo de datos deberá estar lo más simplificado posible.

3. Todas las tablas necesarias para el funcionamiento de la aplicación se generarán automáticamente.

4. La información externa procedente únicamente de fuentes oficiales será recogida de forma automática por la aplicación que se encargará, también, de actualizarla en su base de datos.

5. Toda petición de información se deberá servir en la mayor brevedad de tiempo posible. Para ello será imprescindible la implementación de una memoria caché que se ocupe de los volúmenes grandes de datos.

6. Los datos se servirán dentro de un rango de fechas elegible por el usuario.

7. El usuario podrá escoger qué separador usar para las fechas de inicio y fin.

8. El formato de fecha será el europeo – dd/MM/yyyy.

9. Los datos se servirán para uno, varios o todos los países disponibles en base de datos.

10. Se podrá solicitar una lista que contenga todos los países disponibles para consulta.

11. Se podrá solicitar una lista que contenga todas las fechas disponibles para consulta.

6.3.3 De carácter más General

1. Las peticiones se atenderán en formato JSON.

2. El paralelismo será empleado con frecuencia con el objetivo de acelerar los procesos.

3. Existirán una API rest para la seguridad, una para el acceso y almacenamiento de los datos y una tercera para interactuar con el usuario.

6.4 Requerimientos no funcionales

0001	***Eficiencia en el servicio de los datos***
1.0	*1.0 - Diciembre 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 - Diciembre 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.*

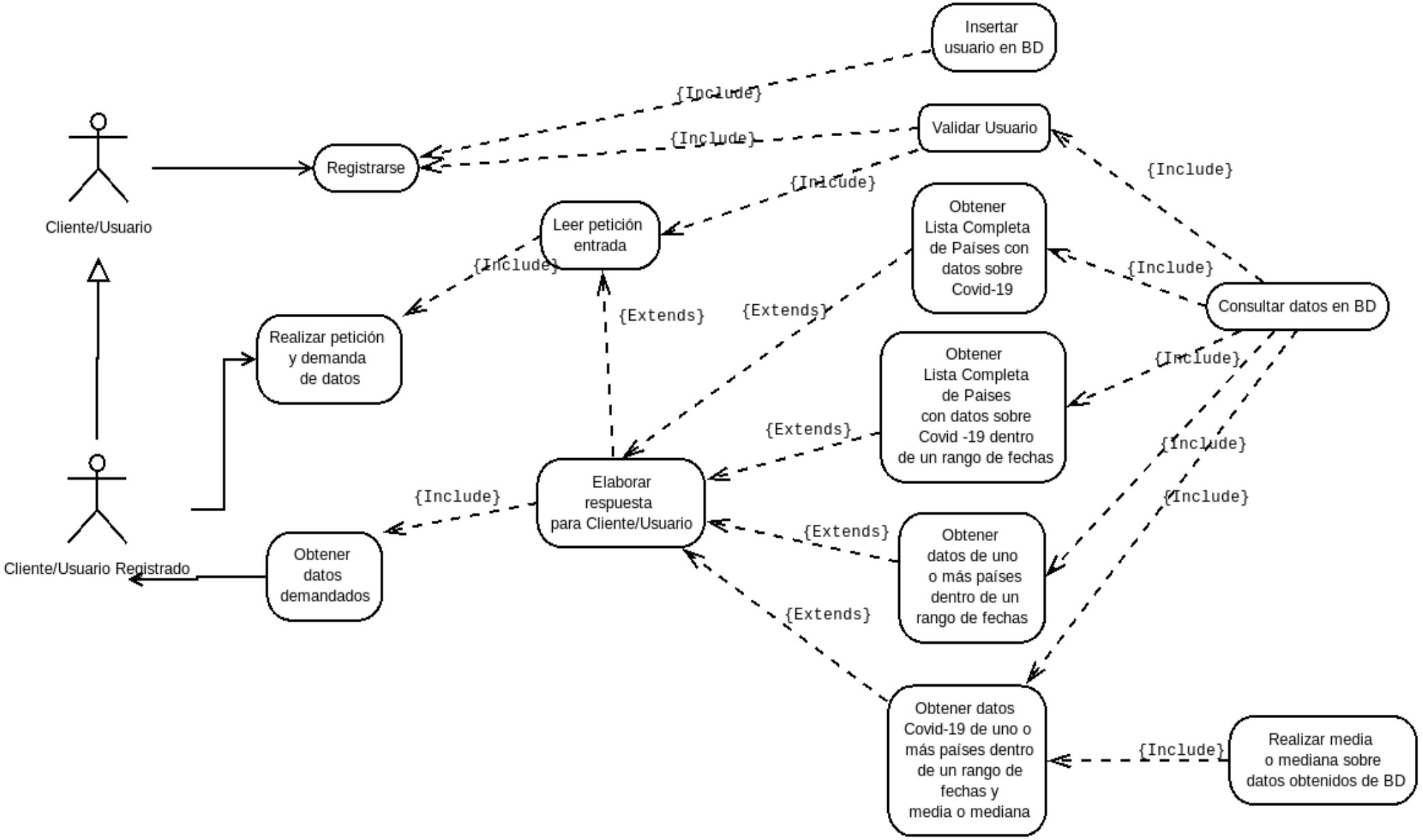
0003	***Seguridad en el almacenamiento de los datos***
1.0	*1.0 - Diciembre 2020*
Dependencias	• *Base de Datos Relacional de la Aplicación* • *.Net Core 3.0 y lenguaje de programación C#* • *PostgreSql* • *Sistema Operativo* • *Técnica de programación*
Descripción	*Los datos sensibles relacionados con 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 - Diciembre 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 - Diciembre 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.*

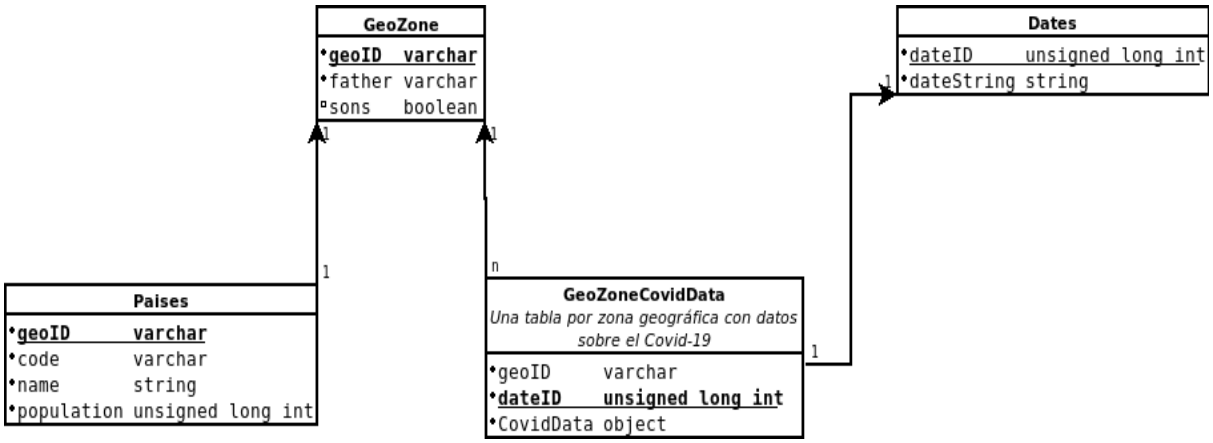
006	***Independencia del SO***
1.0	*1.0 - Diciembre 2020*
Dependencias	• *Base de Datos Relacional de la Aplicación* • *.Net Core 3.0 y lenguaje de programación C#* • *PostgreSql* • *Sistema Operativo* • *Técnica de programación*
Descripción	*La aplicación debe funcionar en cualquier sistema operativo.*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

6.5 Casos de uso



6.6 Modelo Entidad Relacional

6.6.1 Modelo E/R Datos Covid



6.6.2 Modelo E/R Base de Datos Usuarios

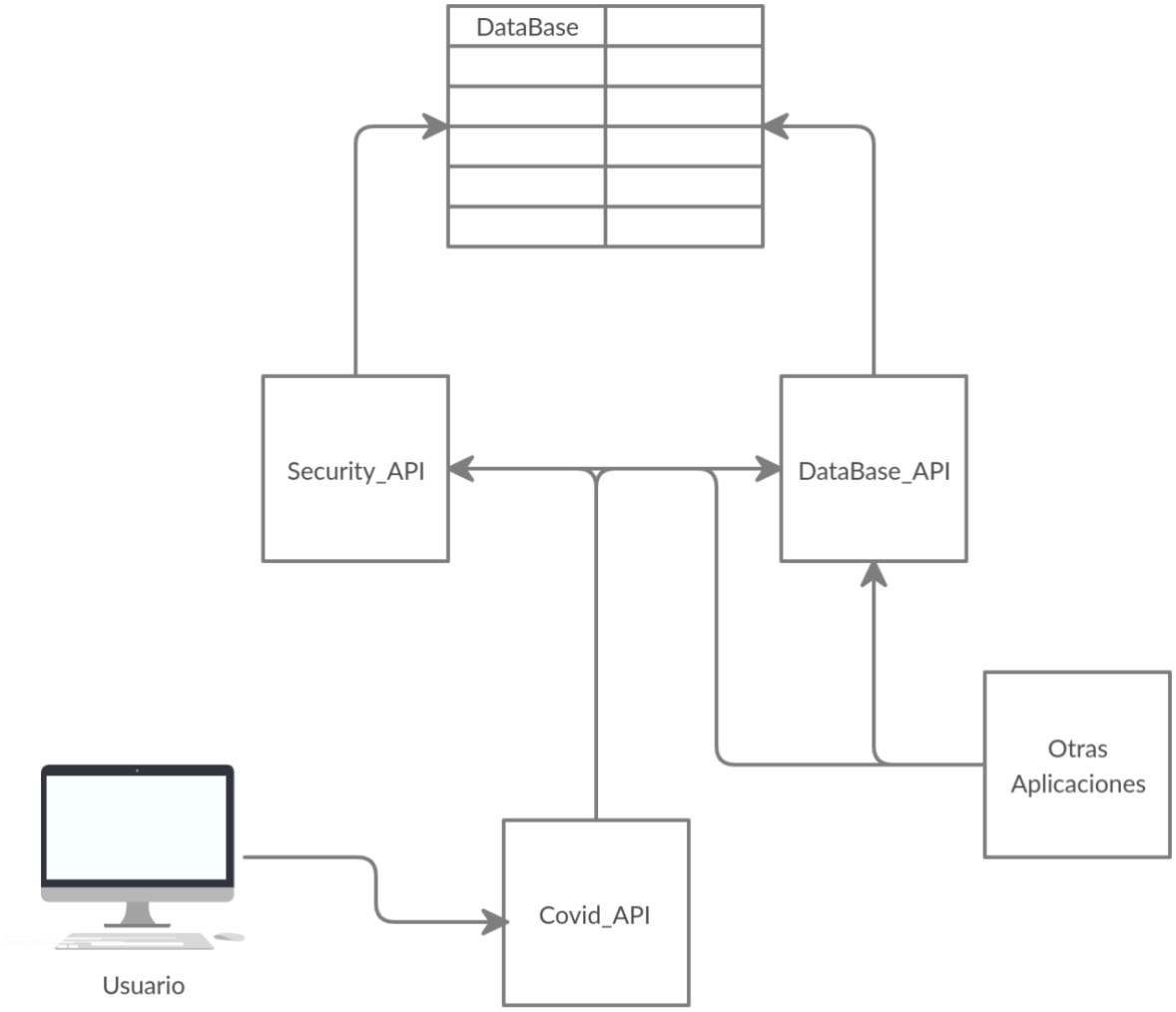
7. Diseño

7.1 Arquitectura

Como ya se ha mencionado, se ha seguido una arquitectura orientada a microservicios. Ésta consiste en una perspectiva para el desarrollo de aplicaciones informáticas mediante la cual son construidas uniendo un conjunto de pequeños servicios que se ejecutan de manera autónoma dentro de su propio proceso. Normalmente se comunican entre sí a través de una API con recursos HTTP. Puede encontrarse más información desde esta url: https://es.wikipedia.org/wiki/Arquitectura_de_microservicios .

Esto quiere decir que cada uno de los servicios que componen la aplicación posee su propia jerarquía de clases y distribución de las mismas por lo que será necesario contemplarlos por separado.

7.2 Diagrama de módulos:



El diagrama anterior muestra una descripción general acerca de cómo está estructurada la aplicación y cómo se relacionan entre sí sus componentes. En él puede apreciarse una de las ventajas de usar la arquitectura orientada a microservicios: otras aplicaciones pueden emplear los componentes para funcionar.

7.3 Descripción jerárquica:

La aplicación se divide en dos servicios componentes, con capacidad para funcionar independientemente y uno que hace las funciones de orquestador del conjunto ofreciéndole sus funcionalidades al usuario. Se percibe, así, que existe un componente dependiente de otros dos para funcionar a pesar de tener la función de orquestarlos.

Así, el microservicio Security_API puede tratarse como una parte constituyente del sistema con capacidad para funcionar independientemente de las demás. Únicamente se relacionará mediante peticiones HTTP con el componente que ofrece sus funcionalidades al usuario/os . En este caso se trata de Covid_API. Por supuesto contará con su propio acceso a una base de datos a determinar, en última instancia por el usuario aunque, es preferible que ésta sea independiente del resto del sistema pues almacenará información más sensible.

Por otro lado, el componente DataAccess_API posee la responsabilidad de almacenar, gestionar, generar y recuperar todos los datos relativos a la pandemia de Covid-19. Únicamente dependerá de una conexión a base de datos pues el despliegue de tablas, realización de consultas, actualizaciones y recogida de los datos es responsabilidad suya y por lo tanto se encuentra automatizado. Se comunicará con el componente que actuará ofreciendo sus servicios al usuario o usuarios Covid_API.

Finalmente el orquestador: Covid_API. Sus funciones son las de actuar como referente de comunicación con el usuario/os ofreciendo las funcionalidades de los dos componentes anteriores. Además, añadirá la responsabilidad de gestión de la propagación de identidad generando los JWT tokens una vez autentificado el usuario correspondiente. También es responsabilidad suya no permitir el acceso a los datos a usuarios no registrados.

7.4 Diagrama de módulos:

7.4.1 Data Access API

7.4.1.1 DataInsertions

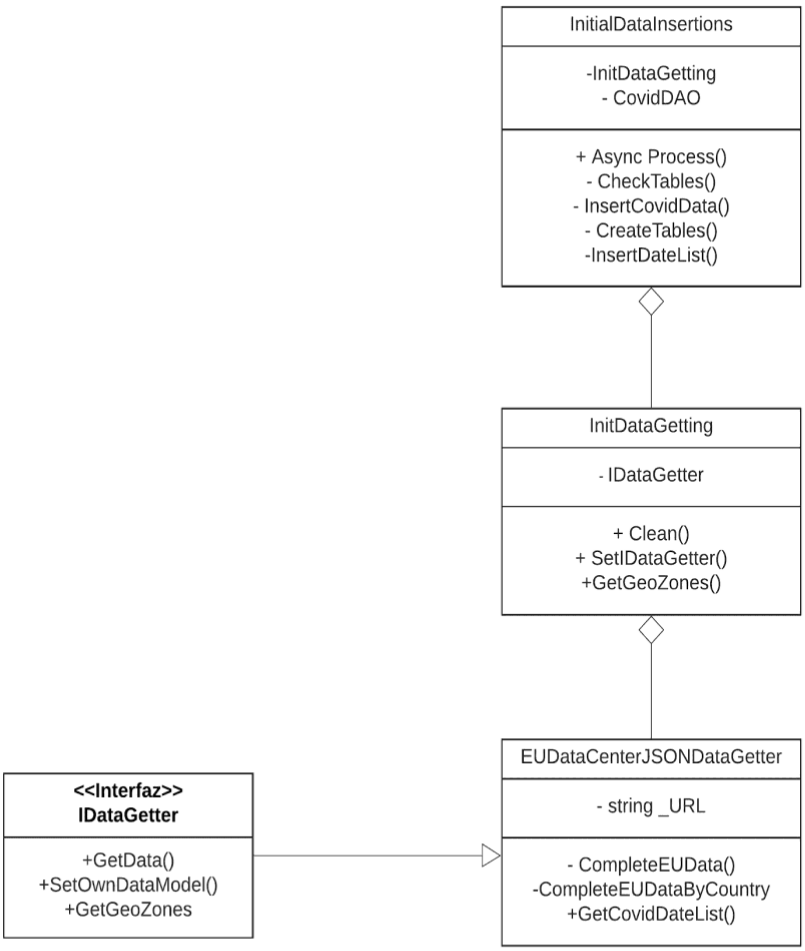
Se trata de un módulo diseñado para gestionar la manipulación y gestión de datos desde el exterior de la aplicación. Sus responsabilidades son:

- Obtención de datos desde diversas fuentes.
- Conversión de los modelos de datos externos al modelo interno.
- Cruce de datos para completar el modelo interno.
- Comprobación de la existencia de tablas necesarias.
- Generación de tablas en base de datos la primera vez que se inicia la aplicación.
- Actualización de datos diaria de manera automática.
- Empleo de concurrencia para liberar el resto de procesos de la aplicación.

El diseño se ha realizado para que sea ampliable en un futuro dependiendo de las diversas fuentes empleadas para obtener los datos. En este caso el centro de datos Europeo ha sido el tomado como base por lo que es la misma clase de objeto la que se encarga de obtener los datos de su proveedor y de completar éstos con los diferentes servicios existentes a nivel global pero, al basarse en una interfaz, en cualquier momento puede generarse una clase de obtenedor de datos distinta que la implemente y componer con ella los niveles superiores que, al fin y al cabo, serán los que se sirvan de ella.

Hay que mencionar que todos los procesos producidos dentro de este módulo se ejecutarán indefinidamente dentro de un hilo distinto al del programa principal. Este hilo será lanzado al poner en marcha la aplicación y quedará en funcionamiento indefinidamente mientras la misma se encuentre actuando. Una vez cumplidas sus funciones quedará en suspenso durante veinticuatro horas, tras las cuales volverá a inicializarse para actualizar la información diaria.

El diagrama de clases es el siguiente:



- Interfaz IDataGetter:

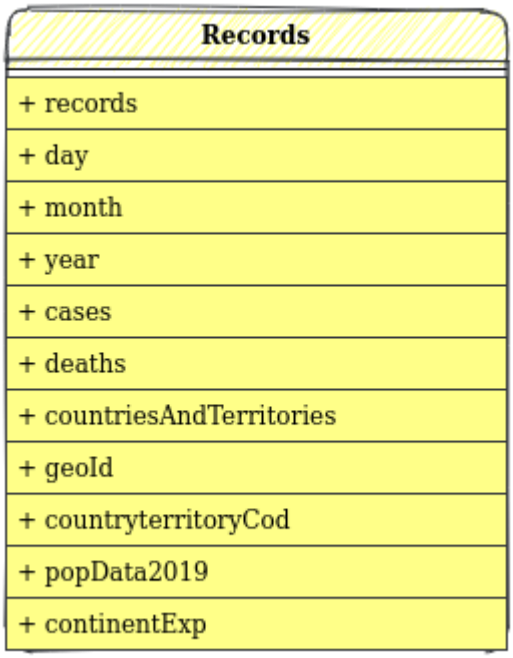
Define los métodos que debe de implementar cualquier clase de objeto encargada de recolectar los datos desde un servidor externo. Son tres los pasos a realizar:

1. Obtener los datos en el formato externo.
2. Convertirlos al formato interno.
3. Servírselos al resto de clases de la aplicación.

- EUDataCenterJSONDataGetter:

Se trata de la clase encargada de implementar la recogida de datos desde un servidor propio de la Unión Europea. El atributo principal es, precisamente, la url desde la que este servicio se encuentra disponible. Incluye tres métodos adicionales a los de la interfaz. Han sido generados para completar la información no disponible entrecruzándola con la de otros servicios disponibles. Además, sirve una lista de fechas disponibles para consulta.

El modelo de datos propio del centro de datos de la UE:



El modelo de datos con el que es completado el anterior:

- InitDataGetting:

Esta clase, compuesta con una interfaz IDataGetter que será su único atributo, tiene como funciones servir los datos obtenidos por la interfaz al resto de la aplicación. Además, será la encargada de liberar la memoria del ordenador una vez esos datos hayan sido insertados en base de datos y, tiene un método que puede variar qué clase de obtenedor de datos será instanciado en su interior como atributo dependiendo el nombre del proveedor de los mismos.

- InitialDataInsertions:

Es el consumidor preferente de los servicios presentados anteriormente y única clase accesible desde el exterior. Su proceso es asíncrono, por lo que se lanza dentro de un hilo independiente del que sigue el resto de la aplicación. Se va a encargar de comprobar la existencia de las tablas necesarias en base de datos creándolas en caso de que no hayan sido generadas con anterioridad.

Gestionará también la recogida de los mismos manejando la clase InitDataGetting y efectuando la inserción en base de datos con la ayuda de la clase DAO correspondiente que será descrita a continuación. Una vez completado su proceso principal – comprobación de tablas, generación de las mismas si corresponde, recolección de datos, persistencia – pasará a un segundo plano hasta cumplidas veinticuatro horas, tras las cuales volverá a iniciarlo manteniendo los datos de la aplicación siempre actualizados.

7.4.1.2 CovidDAO

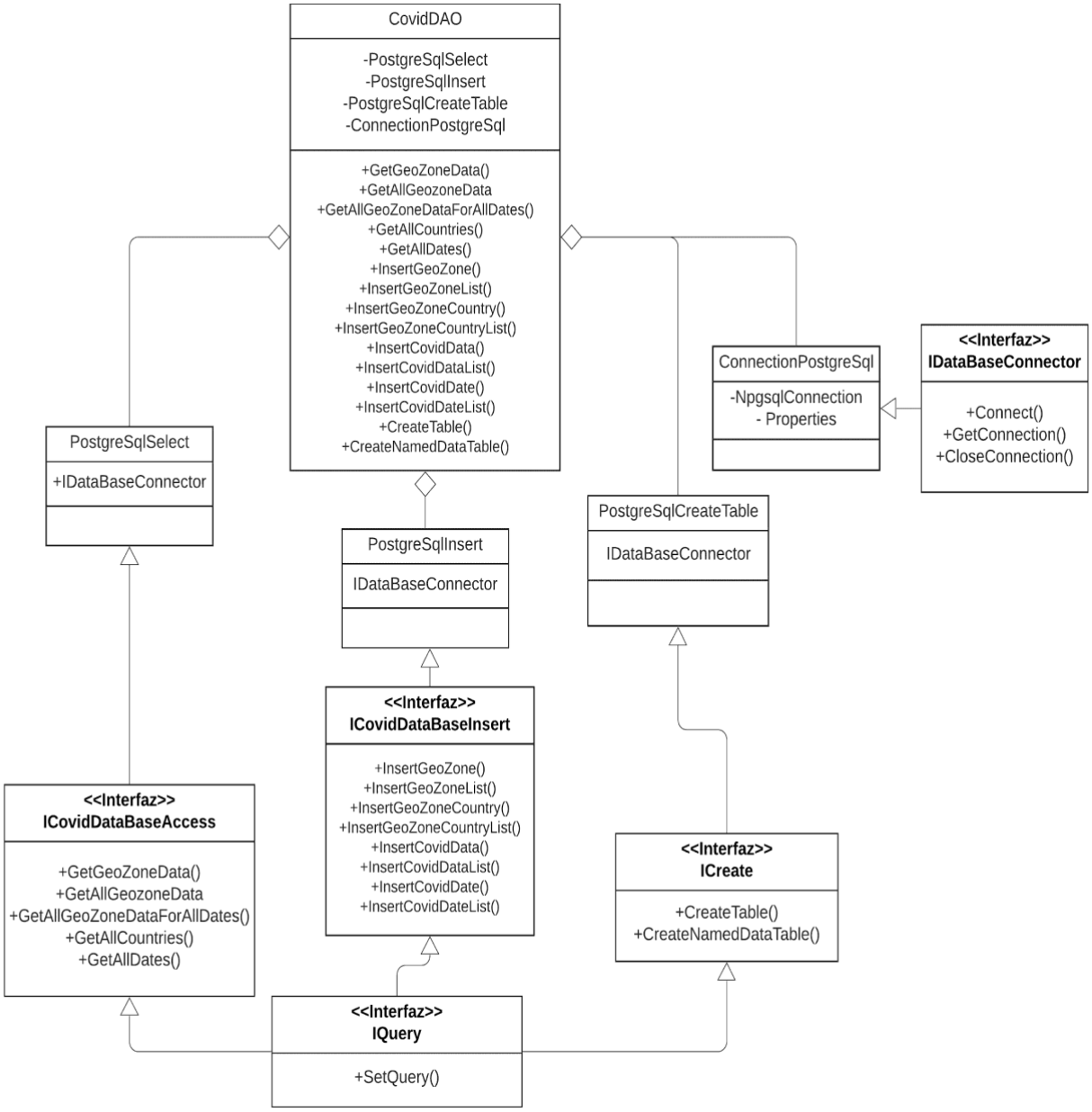
<<Data Access Object>> por sus siglas en inglés. Se trata de un objeto cuya responsabilidad es gestionar el acceso a los datos de la aplicación. Como clase abstracta facilitará el cambio de proveedor y medio de datos dentro de la aplicación pues, en caso de producirse uno o varios cambios simplemente será necesario reimplementar este objeto sin necesidad de modificar el resto de la aplicación.

La de este API se trata de una implementación por composición. La clase abstracta define todos los métodos a emplear que serán adoptados por diferentes interfaces con el objetivo de distinguir entre las operaciones de conexión, creación, inserción y selección de datos.

Cada interfaz resultado del proceso de disgregación la implementará un objeto propio para cada servidor externo de base de datos y operación y serán estos objetos los que compondrán una clase que implementará todos los métodos abstractos de la clase general.

El resultado será una clase mayor con métodos heredados de otra clase abstracta padre que será tratada como esa clase de la que hereda dentro de la aplicación. Así, únicamente sería necesaria la modificación de las instancias de la misma, no su manipulación.

Como se trata de una implementación base no se han usado patrones auxiliares como el de la Factoría que obligaría a generar la clase de objeto esperada dentro de un único gestor por lo que la modificación del servidor de acceso a datos se simplificará notablemente. No obstante, el requisito indispensable mínimo para que su implementación tuviese sentido no se cumple ya que únicamente se ha contemplado un proveedor de acceso a datos.



- IDataBaseConnector:

Es una interfaz que obliga a la implementación de las funcionalidades relativas a la conexión con base de datos: la conexión, la obtención del objeto que gestiona esta por parte del que lo va a utilizar y el cierre de la conexión.

- ConnectionPostgreSql:

Se trata de la implementación de la interfaz de conexión para la base de datos de PostgreSQL. Será uno de los atributos de una clase más general y sus funcionalidades servirán a otras clases para cumplir con sus correspondientes responsabilidades.

- IQuery:

Se trata de una interfaz cuya única funcionalidad consiste en la generación de un objeto consulta empleable por cada uno de sus implementadores. Garantiza que cada implementador sepa generar sus propias consultas dependiendo de los datos recibidos del exterior únicamente en aquellos campos preseleccionados.

- ICreate:

Cualquier clase de objeto encargado de gestionar la creación de tablas deberá de implementar esta interfaz que obligará a la generación de dos funcionalidades: la creación de tablas y la creación de tablas cuyos nombres han sido pasados por parámetro con lo que necesita de la implementación de la interfaz <<IQuery>>.

- PostgresqlCreateTable:

Implementa la interfaz anterior. Necesita de una conexión a base de datos PostgreSQL para funcionar. Va a componer una clase mayor que delegará en ésta las operaciones de creación de tablas por lo que debe de ser autosuficiente a la hora de generar sus propias consultas y componerlas en base a la mínima cantidad de datos recibidos desde el exterior. Para ello dispondrá de su propia ruta hacia las consultas previamente establecidas y almacenadas en formato JSON dentro de su carpeta CreateTablesQueries.

- ICovidDataBaseInsert:

Se trata de una interfaz que compromete a implementar las operaciones necesarias de inserción en base de datos para garantizar la persistencia de la información de la aplicación. La inserción de uno o varios datos referentes a una zona concreta, la inserción de uno o varios países disponibles para consulta así como la inserción de una o varias fechas para la o las que existe/en datos.

- PostgreSQLInsert:

Implementa la interfaz anterior para la base de datos PostgreSQL. Requiere, para su funcionamiento, de un atributo conexión que gestione la misma con base de datos. Es capaz de, en base a los datos de entrada generar sus propias consultas a partir del uso de unas plantillas a las que tiene acceso y han sido preparadas, en formato JSON para su uso exclusivo.

- ICovidDataBaseAccess:

Declara y obliga a la implementación de las operaciones de consulta y recuperación de información desde base de datos. Estas operaciones pueden ser para datos de un país concreto dentro de un rango de fechas, para recuperar los datos de varios países dentro de un rango de fechas, para recuperar la información de todos los países para todas las fechas disponibles, para la recuperación de una lista de países de los que se dispone información o para la recuperación de una lista de fechas para las que existen datos disponibles.

- PostgreSqlSelect:

Implementa la interfaz de acceso a datos para la base de datos PostgreSQL. Requiere de una conexión con la misma como medio de transmisión de consultas. Dichas consultas, como ocurre con el resto de clases de este módulo son accesibles únicamente para esta clase que las cargará desde un archivo JSON.

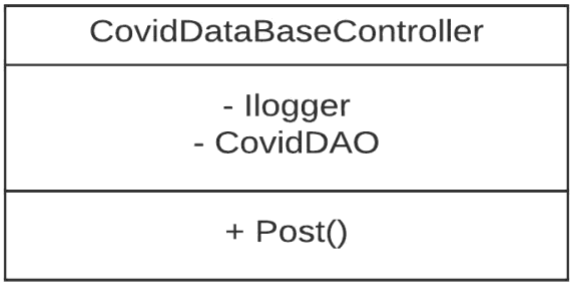
- CovidDAO:

Se trata de una clase abstracta que define todas las operaciones necesarias para comunicarse con la base de datos. A lo largo de toda la aplicación se usará una referencia a ella para ejecutar cualquier consulta que sea requerida por el flujo mismo de ella. Existe una implementación llamada PostgreSqlCovidDAO que está compuesta por cada una de las interfaces operacionales definidas anteriormente. Para cada uno de sus métodos delegará la operación a una de sus clases componentes que será la responsable de la misma.

7.4.1.3 DataAccessControler

Se trata del controlador encargado de recibir las peticiones HTTP destinadas a la aplicación, redirigir la información de entrada hacia el lugar correcto y elaborar la respuesta que los usuarios recibirán.

Diagrama de clase:



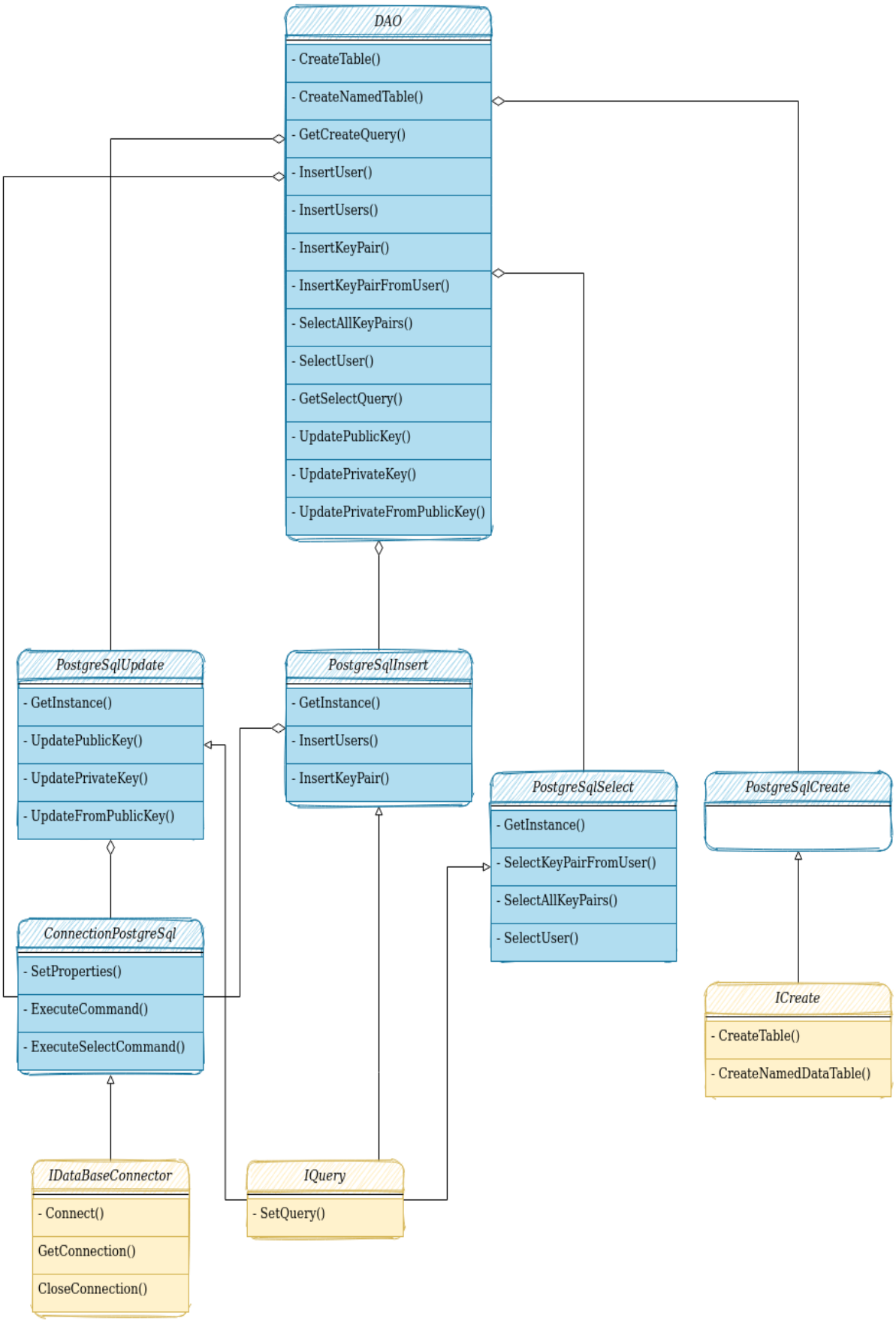
Únicamente dispone de dos atributos, una interfaz gestora de logs y archivos de histórico de operaciones y un CovidDAO que será empleado para acceder a los datos insertados en base de datos.

7.4.2 Security API

7.4.2.1 Security DAO

Al igual que en la anterior parte componente del sistema para la gestión de todo cuanto se encuentre relacionado con la base de datos – creación de tablas, inserción, actualización, borrado y selección de datos de las mismas – es gestionado por un objeto DAO o <<Data Access Object>>. Siguiendo la misma línea este objeto ha sido compuesto por un grupo de clases en cada una de las cuales se delega una responsabilidad. El objetivo es alcanzar la flexibilidad en la gestión de los datos por parte de la aplicación.

En la página siguiente puede observarse un diagrama de clases completo.


DAO
- CreateTable()
- CreateNamedTable()
- GetCreateQuery()
- InsertUser()
- InsertUsers()
- InsertKeyPair()
- InsertKeyPairFromUser()
- SelectAllKeyPairs()
- SelectUser()
- GetSelectQuery()
- UpdatePublicKey()
- UpdatePrivateKey()
- UpdatePrivateFromPublicKey()
PostgreSqlUpdate
- GetInstance()
- UpdatePublicKey()
- UpdatePrivateKey()
- UpdateFromPublicKey()
PostgreSqlInsert
- GetInstance()
- InsertUsers()
- InsertKeyPair()
PostgreSqlSelect
- GetInstance()
- SelectKeyPairFromUser()
- SelectAllKeyPairs()
- SelectUser()
PostgreSqlCreate
ConnectionPostgreSql
- SetProperties()
- ExecuteCommand()
- ExecuteSelectCommand()
ICreate
- CreateTable()
- CreateNamedDataTable()
IDataBaseConnector
- Connect()
GetConnection()
CloseConnection()
IQuery
- SetQuery()

- Interfaz IQuery:

Compromete a la generación de un objeto de consulta propio a la clase que lo implemente.

- Interfaz IDataBaseConnector:

Es una interfaz pensada para definir las operaciones fundamentales de gestión de la conexión a base de datos: conectar, desconectar y obtener el objeto que representa la conexión.

- ConnectionPostgreSQL:

Se trata de la implementación de la interfaz de conexión para la base de datos de PostgreSql. Además de las operaciones relativas a la interfaz es capaz de cargar las propiedades de conexión desde un archivo en formato JSON y de ejecutar más de un tipo de sentencia sql.

- PostgreSQLInsert:

Tiene como responsabilidad la gestión de las operaciones de inserción en una base de datos PostgreSql. Se trata de un objeto del que solamente puede existir una copia dentro de la aplicación y que ofrecerá la posibilidad de insertar nuevos usuarios y nuevos pares de claves privada y pública. Posee como propiedad un objeto ConnectionPostgreSQL para manejar sus consultas y es capaz de establecer las mismas a través de un objeto de consulta que carga desde su propia carpeta contenedora, quedando así restringidas las consultas que realiza.

- PostgreSQLUpdate:

Esta clase maneja las actualizaciones de información de la base de datos. Es un objeto de única copia dentro de la aplicación. Está compuesto por una conexión - ConnectionPostgreSQL – e implementa la interfaz IQuery por lo que es capaz de establecer sus propias consultas en base a la información recibida del exterior y las

plantillas en formato JSON de las que dispone. Puede actualizar una clave pública, una clave privada y un conjunto de datos tomando como base una clave pública.

- PostgreSQLInsert:

Es una clase cuya responsabilidad consiste en la recuperación de la información de usuarios y claves privada y pública de una base de datos PostgreSql. En este caso la conexión es recibida desde el exterior al demandar la consulta esperando así una nueva conexión cada vez que se realiza la llamada a uno de sus métodos. Es capaz de precargar una serie de plantillas con las consultas que realizará desde un archivo JSON situado en su misma ubicación, modificándolas para adaptarlas a la demanda de información de cada situación particular. Puede recuperar de base de datos el par de claves pública y privada de un usuario, todas las claves públicas y privadas existentes en base de datos y la información relativa a un usuario.

- Interfaz ICreate:

Define las operaciones que la clase responsable de la creación de tablas ha de llevar a cabo.

- PostgreSQLCreate:

Es la clase responsable de crear las tablas en base de datos empleando sus propias plantillas de creación. Como este proceso se realiza una única vez ha quedado al margen de los demás y autogestiona completamente la obtención de las tablas a generar desde un archivo propio, los campos de dichas tablas y los nombres de éstas y sus columnas que quedarán lo menos expuestos posible. Recibirá una conexión desde el exterior en el momento de su creación. Sus métodos no serán reutilizables.

- DAO:

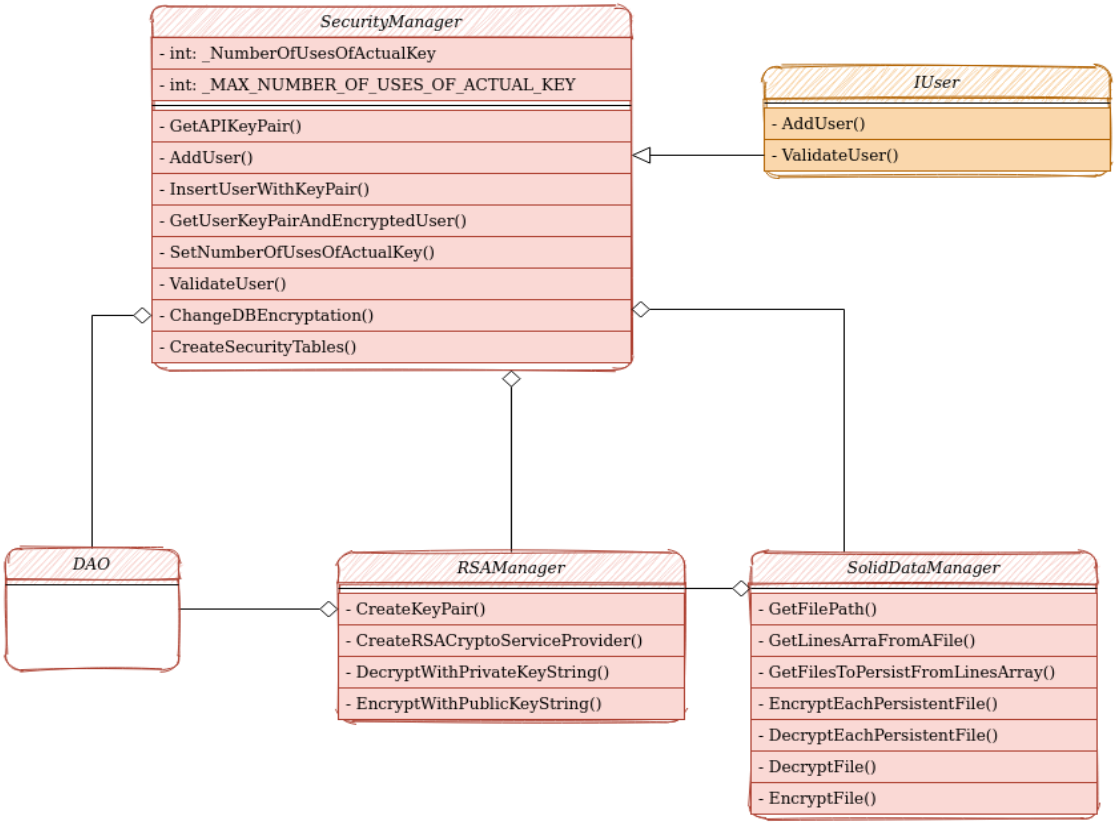
Se trata del objeto que compuesto por todos los anteriores gestiona completamente el acceso a la base de datos de la aplicación. Es una clase abstracta que recibirá una implementación particular para cada gestor de base de datos que decida emplearse, en este caso PostgreSql. Se trata de una abstracción del proceso

de gestión de base de datos por parte de la aplicación garantizando que exista cierta flexibilidad a la hora de modificar el proveedor de los servicios de persistencia.

Sus operaciones coinciden con los métodos de sus clases componentes añadiendo la posibilidad de obtener los objetos de consulta para las operaciones de selección y de creación de tablas y posibilitar la generación de las mismas dentro de un ambiente de campos y propiedades encriptadas.

7.4.2.2 Security Manager

Se trata de una clase que, compuesta por otras tres, gestiona todo el aspecto de la seguridad en el acceso, creación, actualización y validación de datos de usuario así como de la misma dentro de la aplicación. Está pensada para emplear un conjunto de claves pública y privadas que encripten toda la información que la aplicación necesita para su funcionamiento y gestionar la renovación de dicha encriptación así como de desencadenar los procesos de generación de tablas y validación y creación de usuarios.



- RSAManager:

Es un gestor de claves pública y privada y de su empleo. Las crea en caso necesario y encripta a través de una clave pública y desencripta a través de una clave privada. Se vale de un objeto DAO para la gestión, persistencia y el uso de las claves pública y privada internas de la aplicación empleadas como medio de seguridad interna.

- SolidDataManager:

Gestiona la manipulación y lectura de archivos en disco sólido o disco duro valiéndose de un RSAManager para realizar las encriptaciones y desencriptaciones de dichos archivos garantizando la hermeticidad de la información que pueda considerarse sensible dentro de la aplicación. Las plantillas de consulta, el nombre de las tablas empleadas o las propiedades de conexión a base de datos son ejemplos de archivo sensible.

- Interfaz IUser:

Define las operaciones de generación y validación de usuario que serán empleadas por la clase responsable de las mismas.

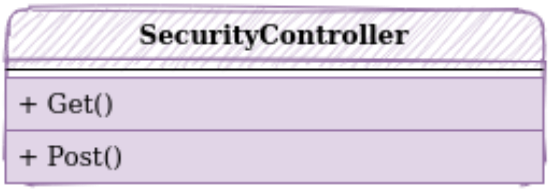
- SecurityManager:

Compuesta por las anteriores y un DAO gestiona todas las operaciones de seguridad de la aplicación, desde la comprobación de la existencia de las tablas y su generación hasta la creación y validación de los usuarios delegando determinadas operaciones en sus clases componentes.

Realiza varios procesos. Una de sus propiedades lleva la cuenta del número de veces que las claves privadas y públicas de uso interno de la aplicación han sido empleadas persistiendo en disco duro esta misma propiedad que se carga desde el mismo al inicializarse la clase. Una vez alcanzado el número máximo desencadena el proceso de cambio de claves y de encriptación de toda la base de datos hacia las nuevas y su sustitución en el registro propio de la aplicación. También inserta nuevos usuarios tras encriptar sus datos y posteriormente los valida y recupera de base de datos las claves pública y privada que serán empleadas para la creación de nuevos usuarios.

7.4.3.3 Security Controller

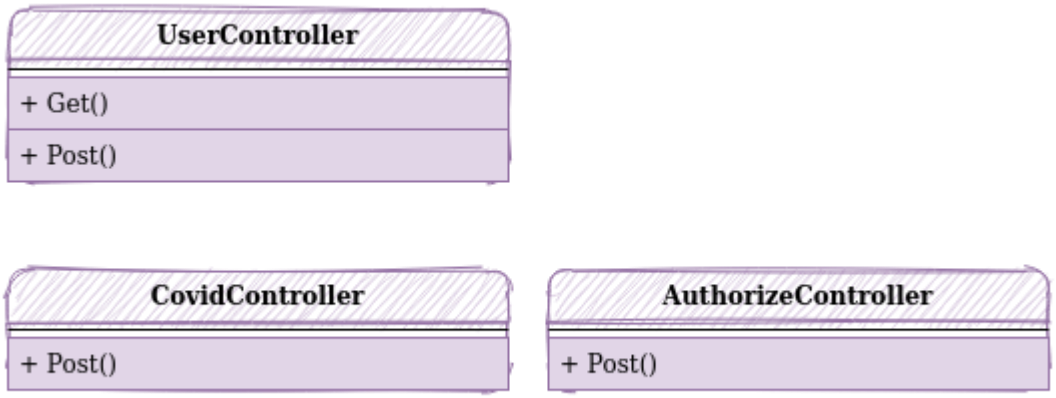
Se trata del controlador generado para gestionar las peticiones HTTP que recibirá la aplicación desde el exterior. El método Get suministra una clave pública con la que encriptar los datos sensibles de los nuevos usuarios y el método Post que recibe los usuarios que van a ser dados de alta o deben de validarse.



7.4.3 Covid API

Es un módulo diseñado únicamente para realizar de interfaz con el usuario. Por medio del mismo este último puede interactuar con los dos módulos ya descritos. Solamente añade un servicio extra de tokenización mediante el cual el usuario obtiene un BearerToken válido durante 24h con el que evitar sucesivas autenticaciones. El resto de servicios son un simple redireccionamiento hacia los módulos que componen la aplicación. Ofrece además la implementación de una memoria caché que favorece la rápida obtención de los datos en peticiones de peso.

Se encargan de recibir las peticiones POST o GET y redireccionarlas al componente correspondiente.



- UserController:

Redirecciona hacia el controlador de usuarios del módulo Security API.

- AuthorizeController:

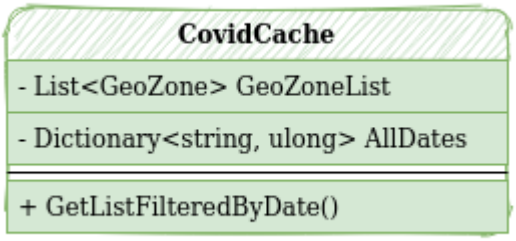
Tokeniza la autenticación empleando para ello el servicio del mismo nombre del módulo Security API.

- CovidController:

Ofrece datos relativos a la pandemia de Covid-19 empleando la memoria caché o el redireccionamiento hacia el microservicio DataAccess API.

7.4.3.2 CovidCache

Extrae el conjunto de los datos disponibles en base de datos y los ofrece de manera rápida filtrando por rango de fecha desde memoria RAM.

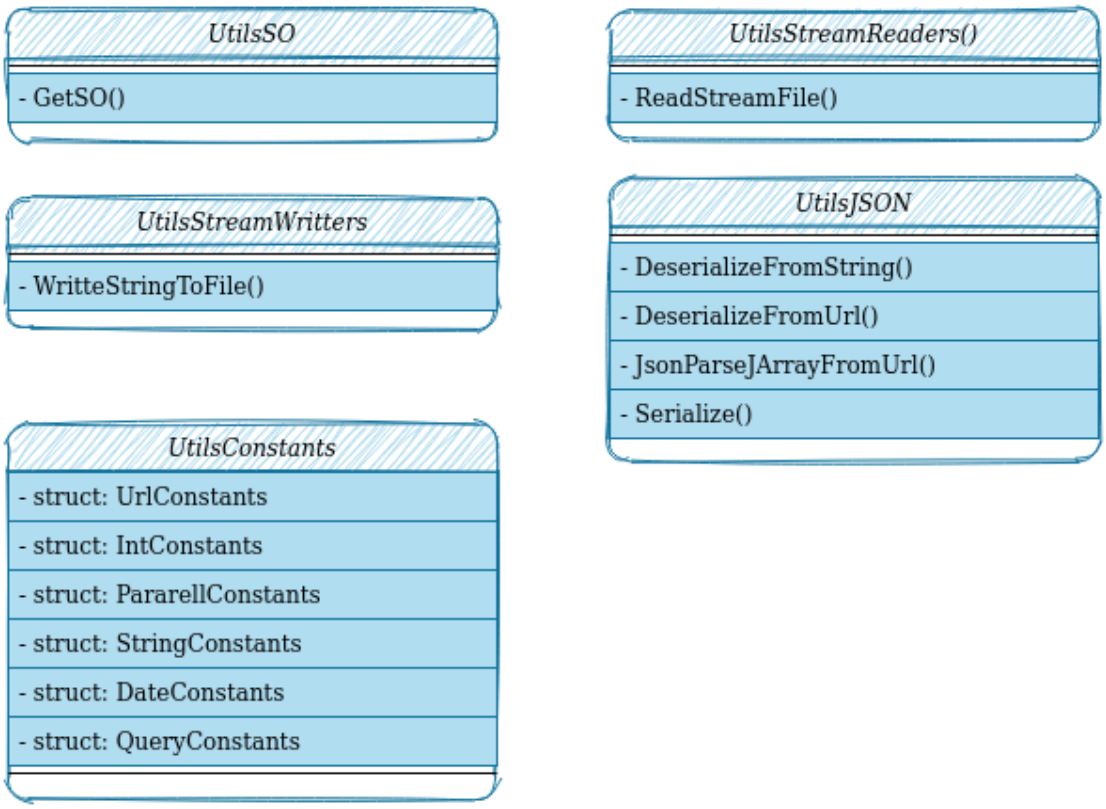


7.4.3.3 TokenGenerator

Cumple las funciones relativas a la generación de tokens y su almacenamiento.

7.4.4 Clases comunes

Existen una serie de clases comunes a todos los servicios. Éstas incluyen funciones de serializado y deserializado de objetos Json desde urls externas, archivos o cadenas de texto, constantes, operaciones de lectura y escritura en disco o reconocimiento del sistema operativo. Éstas clases son las siguientes:



7.4.5 Dependencias Externas

Únicamente se emplea como dependencia externa al propio marco de trabajo de .Net Core una librería llamada Newtonsoft. Ésta es utilizada en la serialización y deserialización de objetos Json por su versatilidad y facilidad. Además, complementa perfectamente la que el marco de trabajo incluye de manera nativa. Su documentación oficial puede encontrarse en:

https://www.newtonsoft.com/json/help/html/Introduction.htm

8. Plan de Contingencia

8.1 Análisis de riesgos

8.1.1 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.

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

8.2 Plan de contingencia

8.2.1 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.

8.2.2 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 preventivas	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 la aplicación	El desarrollador ha de informar. El gestor del proyecto ha de revisar que los tiempos marchan adecuadamente
Falta de seguridad en el tratamiento de los datos	Desconocimiento. Inexperiencia	Media	Información	Refactorizar	El gestor de proyectos ha de revisar meticulosamente como está implementado el proyecto. El arquitecto ha de conocer las leyes vigentes.
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 supervisor del proyecto debe supervisarlo.
Fallo de los sistemas físicos	Mala calidad de los equipos. Imprevisto.	Baja	Equipos de calidad y con garantía.	Reposición de maquinaria estropeada.	Email. Teléfono.

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 Pruebas de software

Para esta aplicación se ha realizado un enfoque orientado al comportamiento. El planteamiento es muy simple: la única prueba válida es la que se le realiza a la aplicación en funcionamiento. Quedan así descartadas técnicas de prueba conocidas como de mockeo y todas aquellas que no ejecuten el algoritmo que compone la aplicación fuera de un caso de uso real de la misma pues poco o nada demuestran.

La ventaja de poseer distintos componentes es que pueden probarse separadamente facilitando la especificación de las pruebas al realizar pues, al centrar el objetivo se previenen desviaciones innecesarias y surgen más puntos a comprobar.

9.3.1 Pruebas al componente de Seguridad

El Método Get para obtener clave pública previa al registro de usuario funciona.	**001**	**Obligatoria**
	Prueba de funcionamiento	**Sí**
Descripción: Se han de realizar varias peticiones Get a la url identificativa del servicio. Éstas deben devolver una clave pública con la que encriptar la información de usuario antes de solicitar un alta. La clave debe de ser la misma para los primeros X usos.		
Prerrequisitos: Haber instalado todas las dependencias. Poner en marcha el componente Security_API de la aplicación.		
Pasos:	**Datos**:	
Introducir url	Url	
Obtener respuesta	Clave Pública de Aplicación	
Repetir proceso		

Registro de usuario satisfactorio	**002**	**Obligatoria**
	Prueba de funcionamiento	**Sí**

Descripción:

Se realizará el registro de varios usuario, con mismo y distinto correo electrónico. En caso de estar registrado se comprobará la actualización de los datos. En caso de no estarlo, se recibirá una clave pública única para cada usuario.

Prerrequisitos:

Haber instalado todas las dependencias. Poner en marcha el componente Security_API de la aplicación. Obtener la clave pública de encriptado de la aplicación mediante método Get.

Pasos:	**Datos**:
Obtener Clave Pública de Aplicación.	Clave pública
Encriptar los datos de usuario con la clave pública obtenida.	Modelo usuario
Establecer el elemento new del modelo de usuario a true.	Modelo usuario.
Enviar petición POST	Modelo de usuario.Url
Recibir respuesta	Clave pública.
Repetir proceso	

Validación de usuario	**003**	**Obligatoria**
	Prueba de funcionamiento	**Sí**
Descripción: Con un grupo de usuarios creados con anterioridad se realizará su autentificación obteniendo, true en caso afirmativo y false en caso negativo.		
Prerrequisitos: Haber instalado todas las dependencias. Poner en marcha el componente Security_API de la aplicación. Poseer un grupo de usuarios registrados cada uno con su clave pública única.		
Pasos:	**Datos**:	
Encriptar los datos de usuario con clave única	Clave pública. Modelo usuario.	
Establecer el elemento new del modelo de usuario a false.	Modelo usuario.	
Añadir clave pública única al modelo de usuario.	Modelo usuario. Clave pública.	
Enviar petición POST.	Modelo de usuario.Url	
Recibir respuesta	Clave pública.	
Repetir proceso		

La clave pública de aplicación varía cada cierto número de usos así como la encriptación en base de datos de los datos de usuario.	**004**	**Obligatoria**
	Prueba de funcionamiento	**Sí**

Descripción:

Se realizará la petición de obtención de clave pública de aplicación y/o se registrarán o autentificarán usuarios tantas veces como sean necesarias para que se produzca el cambio de claves privada y pública de aplicación. Por defecto está establecido que sean 100.

Prerrequisitos:

Haber instalado todas las dependencias. Poner en marcha el componente Security_API de la aplicación.

Pasos:	**Datos**:
Obtener Clave Pública de Aplicación y/o registrar o autentificar usuario.	Modelo usuario. Clave pública.
Repetir tantas veces como sea necesario.	Modelo usuario. Clave pública.
Comprobar que la respuesta de Obtención de clave pública ha variado.	Clave pública.
Comprobar que en base de datos se han producido las actualizaciones necesarias.	Modelo de usuario.Conexión a base de datos.
Repetir proceso	

9.3.2 Pruebas al componente de acceso a datos

Devuelve el pais solicitado.	**001**	**Obligatoria**
	Prueba de funcionamiento	**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: Haber instalado todas las dependencias. Poner en marcha el componente DataAcccess_API de la aplicación. Haber esperado el tiempo suficiente para que los datos hayan sido insertados en base de datos.		
Pasos:	**Datos**:	
Crear Petición introduciendo el código ISO2 de país en el campo correspondiente. Es necesario introducir, también, el nombre del método correspondiente al que se dirige la aplicación.	Modelo POST	
Enviar Petición	Modelo POST	
Recibir Respuesta	Modelo de salida de datos	
Comprobar Respuesta	Modelo de salida de datos	
Repetir proceso		

Devuelve Países Por Fecha Solicitados	**002**	**Obligatoria**
	Prueba de funcionamiento	**Sí**
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: Haber instalado todas las dependencias. Poner en marcha el componente DataAcccess_API de la aplicación. Haber esperado el tiempo suficiente para que los datos hayan sido insertados en base de datos.		
Pasos:	**Datos**:	
Crear Petición introduciendo una lista de códigos ISO2 de país en el campo correspondiente del modelo. Es necesario introducir, también, el nombre del método correspondiente al que se dirige la aplicación.	Modelo POST	
Enviar Petición	Modelo POST	
Recibir Respuesta	Modelo de salida de datos	
Comprobar Respuesta	Modelo de salida de datos	
Repetir proceso		

Devuelve datos para todos los países disponibles dentro de un rango de fechas	**003**	**Obligatoria**
	Prueba de funcionamiento	**Sí**

Descripción:
Se almacenarán todos los países disponibles en base de datos y, posteriormente se enviarán 100 peticiones que solicitarán datos para todos los países dentro de un rango de fechas. Se comprobará que existen datos para todos los países y que esos datos estarán dentro del período solicitado.

Prerrequisitos:

Haber instalado todas las dependencias. Poner en marcha el componente DataAcccess_API de la aplicación. Haber esperado el tiempo suficiente para que los datos hayan sido insertados en base de datos.

Pasos:	**Datos**:
Solicitar listado de países introduciendo el nombre del método adecuado dentro del campo apropiado de la petición POST.	Listado de países
Crear Petición	Modelo POST
Enviar Petición	Modelo POST
Recibir Respuesta	Modelo de salida de datos
Comprobar Respuesta	Modelo de salida de datos
Repetir proceso	

Devuelve Listado de países	**004**	**Obligatoria**
	Prueba de funcionamiento	**Sí**

Descripción:
Se realizarán 100 peticiones del listado completo de países.

Prerrequisitos:

Haber instalado todas las dependencias. Poner en marcha el componente DataAcccess_API de la aplicación. Haber esperado el tiempo suficiente para que los datos hayan sido insertados en base de datos.

Pasos:	**Datos**:
Crear Petición	Modelo POST
Enviar Petición	Modelo POST
Recibir Respuesta	Modelo de salida de datos
Comprobar Respuesta	Modelo de salida de datos
Repetir Proceso	

Devuelve Listado de fechas disponibles	**005**	**Obligatoria**
	Prueba de funcionamiento	**Sí**
Descripción: Se realizarán 100 peticiones del listado completo de fechas. Se comprobarán su coincidencia a contar desde el primer día de registro de las mismas hasta el día de realización de la prueba.		
Prerrequisitos: Tener en pleno funcionamiento la aplicación.		
Pasos:	**Datos**:	
Crear Petición	Modelo POST	
Enviar Petición	Modelo POST	
Recibir Respuesta	Modelo de salida de datos	
Comprobar Respuesta	Modelo de salida de datos	
Repetir Proceso		

9.3.3 Pruebas de Conjunto

Repetir pruebas realizadas en el componente de Seguridad	**001**	**Obligatoria**
	Prueba de funcionamiento	**Sí**
Descripción: Se repetirán las pruebas realizadas al componente de seguridad, esta vez, empleando la url del componente Covid_API que gestiona el conjunto.		
Prerrequisitos: Haber instalado todas las dependencias. Poner en marcha todos los componentes de la aplicación. Haber cumplido los prerrequisitos de cada uno de ellos.		
Pasos:	**Datos**:	
Levantar todos los componentes	Modelo POST	
Repetir Pruebas para cada componente usando el modelo de datos propio de Covid_API.	Modelo POST	
Recibir Respuesta	Modelo de salida de datos	
Comprobar Respuesta	Modelo de salida de datos	
Repetir Proceso		

Recibir Token JWT tras autenticar un usuario registrado	**002**	**Obligatoria**
	Prueba de funcionamiento	**Sí**
Descripción: Se probará a identificar un conjunto de usuarios previamente registrados y cada vez que la identificación sea exitosa se recibirá un JWT token como respuesta. En caso negativo se mostrará un error HTTP 401.		
Prerrequisitos: Haber instalado todas las dependencias. Poner en marcha todos los componentes de la aplicación. Haber cumplido los prerrequisitos de cada uno de ellos. Poseer varios usuarios registrados.		
Pasos:	**Datos**:	
Enviar petición de autorización a la url correspondiente.	Modelo POST de Usuario.	
Comprobar que la respuesta recibida posibilita la petición de datos al introducir el JWT token en el correspondiente apartado de cabecera.	Modelo POST de Usuario y de CovidData.	
Repetir todo el proceso para cada uno de los usuarios.		

Repetir pruebas del componente de acceso a datos	**003**	**Obligatoria**
	Prueba de funcionamiento	**Sí**
Descripción: Se repetirán todas las pruebas de acceso a datos esta vez empleando el modelo y url propios del componente Covid_API encargado de interactuar con el usuario y gestionar el conjunto.		
Prerrequisitos: Haber instalado todas las dependencias. Poner en marcha todos los componentes de la aplicación. Haber cumplido los prerrequisitos de cada uno de ellos. Haber obtenido un token de acceso tras haber identificado un usuario.		
Pasos:	**Datos**:	
Enviar petición de autorización a la url correspondiente.	Modelo POST de Usuario.	
Obtener token de autenticación JWT.	Modelo POST de Usuario y de CovidData.	
Repetir todo el proceso para cada uno de los usuarios.		

10. Análisis de recursos y costes

10.1 Coste según fase del proyecto

10.1.1 Análisis

Durante esta fase el analista contratado realiza un análisis completo de los requerimientos de la aplicación y se corrige sus posibles puntos débiles.

Responsable	Coste por hora	Número total de días	Actividad	Coste Total
Analista	20 euros	3	Análisis	480 euros
Analista	20 euros	1	Revisión y presentación	160 euros

10.1.2 Diseño

En la fase de diseño el arquitecto de software plasma la estructura de la aplicación. Una vez realizada la primera aproximación itera sobre su diseño con el objetivo de mejorarlo.

Responsable	Coste por hora	Número de días	Actividad	Coste Total
Arquitecto	25 euros	3	Diseño inicial	600 euros
Arquitecto	25 euros	1	Duscusión acerca del diseño inicial	200 euros
Arquitecto	25 euros	1	Correcciones	200 euros

10.1.2 Implementación de Base de Datos

En esta fase el desarrollador especializado en bases de datos diseña e implementa la persistencia de los datos y su modelo.

Responsable	Coste por hora	Número de días	Actividad	Coste Total
Desarrolador base de datos	18 euros	2	Diseño	288 euros
Desarrolador base de datos	18 euros	2	Generación	288 euros

10.1.3 Implementación API

Durante esta fase, la más larga, el desarrollador programa el código que previamente ha sido diseñado por el arquitecto.

Responsable	Coste por hora	Número de días	Actividad	Coste Total
Desarrollador	18 euros	16	Desarrollo de DataAccess_API	2.304 euros
Desarrollador	18 euros	16	Desarrollo de Security_API	2.304 euros
Desarrollador	18 euros	7	Desarrollo Covid_API	1.008 euros

10.1.4 Pruebas de Conjunto

Tanto el desarrollador de base de datos como el desarrollador de aplicaciones realizan una fase de pruebas de la aplicación para comprobar que todo funciona adecuadamente.

Responsable	Coste por hora	Número de días	Actividad	Coste Total
Desarrollador Base de Datos	18 euros	1	Pruebas a base de datos	144 euros
Desarrollador	18 euros	1	Pruebas de la aplicación	144 euros

10.1.5 Subsanación de posibles errores

Tras haber realizado los tests pertinentes es necesario corregir todo cuanto no funcione apropiadamente. En esta fase se realizan esas correcciones.

Responsable	Coste por hora	Número de días	Actividad	Coste Total
Desarrollador Base de Datos	18 euros	2	Corrección de errores	288 euros
Desarrollador	18 euros	2	Corrección de errores	288 euros

10.1.6 Supervisión del proyecto

Durante la realización del proyecto el gestor del mismo ha de responsabilizarse de la buena marcha del mismo. Para asegurarse de ello lo supervisa.

Responsable	Coste por hora	Número de días	Actividad	Coste Total
Gestor de proyectos	30 euros	58	Supervisión y gestión del proyecto	13.920 euros
Gestor de proyectos	30 euros	2	Muestra del proyecto	480 euros

10.1.7 Coste total del proyecto según la fase del mismo

Fase del Proyecto	Coste total	Coste acumulado
Análisis	640 euros	640 euros
Diseño	1000 euros	1640 euros
Implementación base de datos	576 euros	2216 euros
Implementación API	5616 euros	7832 euros
Pruebas de conjunto	288 euros	8120 euros
Muestra	480 euros	8600 euros
Subsanación de posibles errores	1152 euros	9752 euros
Supervisión proyecto	13440 euros	23192 euros
	Total:	**23192 euros**

10.2 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
Servicio Azure (Media) * 1 año	(11.697 + 67.717) / 2 = 39.707 euros	476.484 euros	6641.85 euros
Servicio AWS Base de datos 1 año	90 euros	1080 euros	7721.85
		Total:	**7721.85 euros**

10.3. Coste total del proyecto

Tipo de coste	Coste
Fases del proyecto	23192 euros
Recursos materiales	7721.848 euros
Total	**30913.848 euros**

11. Evaluación y cierre del proyecto

11.1 Cómo será evaluado el proyecto.

El proyecto se evaluará, en primera instancia, por su adecuado funcionamiento. Al tratarse de una aplicación informática la parte técnica debe de considerarse como principal característica. Es decir, que cumpla con la función que desempeñará como herramienta. Por ello no es posible dar este proyecto por terminado hasta que la aplicación haya pasado cada uno de las pruebas de funcionamiento diseñadas para ella.

El segundo criterio de evaluación será la presentación a posibles usuarios de la misma. Puesto que son éstos para quienes estará realizada, su opinión y convencimiento acerca del empleo de la misma ha de regir tanto las funcionalidades finales que ésta presentará como el uso de las mismas. Por lo tanto, no será posible dar por cerrado este proyecto hasta que al menos un usuario decida que la aplicación es lo suficientemente buena para ser usada.

El tercer y último criterio será la propia opinión del equipo que lo ha realizado. Una vez haya unanimidad acerca del grado de perfeccionamiento de la misma podrá darse por finalizado.

11.2 Aspectos a evaluar

11.2.1 Aspectos Funcionales

Aspecto a evaluar	**Criterio de evaluación**	**Nivel de importancia**
La aplicación realiza todas y cada una de las funcionalidades para la que ha sido diseñada.	Las pruebas de funcionamiento pasan adecuadamente.	Máxima.
La aplicación realiza todas y cada una de las funcionalidades de manera eficiente.	Se realizará un estudio adecuado del coste algorítmico de cada uno de los procesos. También se revisará durante la prueba de la misma que no haya ninguna funcionalidad que genera sensación de lentitud a cualquier usuario.	Muy alta.

11.2.2 Aspectos de usuario

Aspecto a evaluar	Criterio de evaluación	Grado de importancia
El usuario comprende con rapidez cómo se emplea.	Se realizará una presentación de la misma a cada usuario y se le consultará acerca de su facilidad de empleo.	Máxima.
El usuario está convencido de su utilidad.	Se le pedirá a cada usuario que realice un pequeño cuestionario para comprobar su grado de satisfacción.	Muy alta.

11.2.3 Aspectos de equipo

Aspectos a evaluar	Criterio de evaluación	Grado de importancia
El equipo piensa que el resultado es satisfactorio.	Se le pedirá a cada uno de los miembros que evalue los aspectos positivos y negativos de la aplicación.	Muy alto.
El equipo está convencido de que cada funcionalidad implementada ha sido adecuadamente realizada.	El equipo expondrá su opición en una reunión de conjunto.	Alto.

12. Conclusión y opinión personal del proyecto

Todo proyecto, tanto personal como de equipo requiere una suma de esfuerzos incuestionable. Eso quiere decir que éste ha costado una gran cantidad de tiempo y esfuerzo. Durante todo ese tiempo existen altibajos y dudas acerca de la motivación del mismo. Siempre se desea que ese sacrificio fructifique como resultado. En este caso el resultado esperado suele ser el beneficio económico. Sin embargo, el beneficio final no depende únicamente del trabajo realizado ni del producto final. Es necesario contar con el entorno y éste no tiene por qué ser favorable ni, aunque sí lo fuese sostener el éxito de ninguna empresa. De hecho, la motivación económica no ha podido ser el objetivo de este proyecto y eso ha supuesto una importante fuente de desmotivación a lo largo de todo el recorrido. Es por ello que se ha hecho necesario el mantenimiento de una actitud positiva y el estar dispuesto a buscar el bien incluso cuando se piensa que no existe alguno. Y he de mencionar que el mayor bien que puede extraerse siempre de cualquier acción es la experiencia y el desarrollo personal. También en este caso.

Por la propia naturaleza de este proyecto podemos deducir que el beneficio que aporte, si es económico, no se producirá salvo indirectamente a través del prestigio o la publicidad que pueda obtener de él la empresa que lo abre al público. Pero aportará un beneficio para todo aquel que emplee el producto final pues elimina la necesidad de crear, gestionar y mantener un programa semejante. Además, al ofrecer la información de forma escogida por el usuario, todo cálculo u operación computacional que deja de realizarse supone un ahorro energético global que, si bien es mínimo, contribuye a reducir emisiones innecesarias de CO2 a la atmósfera aunque sea modestamente. Siguiendo esa cadena de consecuencias, el haber estado inmerso en este proyecto durante todo su desarrollo ha provocado la obligación y la necesidad de abrir la mente a un mundo más amplio e interconectado de lo que podía haber pensado. El hecho de que el open source exista y sea un movimiento que obtiene cada vez más voluntarios permite la posibilidad de realizar programas, y también usarlos de manera gratuita. Y esos programas son perfectamente válidos siempre y cuando posean una documentación bien realizada y una comunidad detrás que las respalde. Cualquier pequeño esfuerzo cuenta, porque se suma a otros muchos y genera la libertad de escoger, no solamente de acceder, a tecnologías que de otro modo estaría fuera del alcance de los individuos. Y eso ha motivado que haya crecido en conocimiento del ambiente tecnológico mundial y aprecie mucho más de lo que ya lo hacía tanto la gratuidad de la tecnología como la adecuada documentación de la misma.

Como conclusión final cabe mencionar que no toda actividad debe de fructificar como un beneficio económico. En el ámbito relacionado con las Ciencias de la Computación, una aplicación puede generar un beneficio social inmenso cuyo volumen es incrementado a través de las redes sociales, internet y un mundo que más pronto que tarde acabará por convertirse en una gran red de personas interconectadas entre sí a través de una máquina. Por eso es de suma importancia que existan proyectos que estén dedicados a permitir que otras empresas se realicen, sin importar el beneficio económico pues éste, si ha de producirse, lo hará gracias a el camino abierto por otras actividades en apariencia, poco provechosas para quien las realiza. Siguiendo esta lógica propia y personal de pensamiento, que es, además, compartida por una gran cantidad de personas que colaboran con proyectos propios, bien ofreciéndolos gratuitamente, bien permitiendo que la totalidad de su código pueda ser copiado de un lugar a otro ahorrando horas de tedioso trabajo, pienso sinceramente que esta aplicación es un aporte sencillo pero versátil para todo aquel que necesite ofrecer ciertos servicios a través de internet y que no disponga del tiempo para desarrollarlos por sí mismo ni quiera mantenerlos.

13. Bibliografía y enlaces de interés

13.1 C#

- Katrib, Miguel y Grupo WEBOO. **Empiece a programar. Un enfoque multiparadigma con C#**. ISBN: 9781973359470.
- Deitel, Paul j. y Deitel, Harvey M.. **C#. Cómo programar**. 2ª Edición. Pearson. Prentice Hall. ISBN: 970-26-1056-7.
- **Documentación ofical .Net** : https://docs.microsoft.com/en-us/dotnet/fundamentals/

13.2 Docker

- Kane, Sean P. y Matthias , Karl. **Docker. Up & Running**. O'Reilly 2ª Edición. ISBN: 978-1-492-03673-9.
- **Documentación oficial**: https://docs.docker.com/reference/

13.3 Postgre SQL

- **Documentación oficial**: https://www.postgresql.org/docs/

13.4 Diseño de software

- Vlissides , John; Helm, Richard; Johnson, Ralph y Gamma, Erich. **Dessign Patterns. Elementes of reusable Object-Oriented software**. Adison-Wesley professional. ISBN: 0-201-63361-2.
- Shvets, Alexander. **Sumérgete en los patrones de diseño**. Autopublicación. Sitio oficial: https://refactoring.guru

13.5 Metodologías

- Stellman , Andrew and Greene, Jennifer. **Head First Agile: A Brain-Friendly Guide to Agile principles, ideas and real world practices**. 1st Edition. ISBN: 9781449314330