Articulo técnico Vacunas.uy

Nicolás San Martín 5.055.041-9, Matías Borggio 4.903.997-1, Ignacio Otero 5.266.715-7, Agustín Ruíz Díaz 5.042.744-4, and Bruno Pardiñas 5.205.075-6

Facultad de Ingeniería - Universidad de la República
J. Herrera y Reissig 565, Montevideo, Uruguay
http://www.fing.edu.uy/

Resumen La motivación de este proyecto es crear un sistema informático web para la gestión de reservas de vacunación para la población Uruguaya, en marco de la pandemia de COVID-19, con el fin de facilitar la organización y la logística de la vacunación. En este artículo se presenta el diseño e implementación de una plataforma informática utilizando Jakarta EE.

Keywords: Jakarta EE · Servicios Web · Computación en la Nube· Android · JSF.

1. Introducción

Este trabajo se plantea en el marco de la edición 2021 de Taller de Sistemas de Empresariales.

La pandemia del Coronavirus está afectando a la población de Uruguay desde hace ya más de un año. Este evento a nivel mundial, que únicamente en nuestro país ha tenido un enorme impacto de manera directa en muchas personas y de manera indirecta en toda la población, afectando no solamente la situación sanitaria, sino también la económica y social. Considerando que este tipo de situaciones puede repetirse en un futuro con otras enfermedades, se están desarrollando soluciones enfocadas en su seguimiento y control. En particular, es de interés abordar una etapa muy importante de este ciclo que es la inoculación de la población, por ejemplo, mediante la aplicación de vacunas.

Para la gestión de todo este proceso, es necesaria la creación de una plataforma que brinde soporte con el fin de facilitar la organización y la logística de la vacunación.

Esta solución debe proveer un sitio web en el cual los usuarios puedan agendar vacunaciones y obtener información sobre los planes de vacunación. Además se debe construir una aplicación móvil android para que los usuarios puedan obtener información desde la misma.

El resto del documento se organiza de la siguiente manera. La sección 2 presenta el Marco Teórico necesario para comprender algunos aspectos de la solución. En la sección 3 una descripción del problema más profunda. En la sección 4 se presenta la solución. En la sección 5 se presenta la Arquitectura del sistema. En la sección 6 se habla de algunos detalles de implementación. En

la sección 7 se evalúa la solución. En la sección 8 se comenta un poco sobre el desarrollo del proyecto. Por último, en la sección 9 se presentan las conclusiones y el trabajo a futuro del proyecto.

2. Marco Teórico

En esta sección se introducen resumidamente los conceptos que son relevantes para la comprensión del artículo.

2.1. [Aplicación Empresarial]

Una Aplicación Empresarial es una aplicación de software desarrollada para administrar las operaciones, activos y recursos de una empresa. En el mundo actual estas aplicaciones frecuentemente trabajan en conjunto con otras aplicaciones desarrolladas de forma independiente. Este concepto es conocido como Integración de Aplicaciones Empresariales. Históricamente se han utilizado distintos enfoques para la integración, tales como, mensajería, RPC, RMI, web services, etc. Para el desarrollo de estas aplicaciones existen distintas plataformas para facilitar y uniformizar el desarrollo e integración brindando soluciones a los problemas típicos. Los ejemplos mas conocidos son Java EE y .net.

2.2. [Cloud Computing]

La computacion en la nube consiste en ofrecer servivios alojados a traves de internet. Permite a las empresas consumir recursos de informatica como una utilidad sin tener que construir y mantener infrastructuras en la empresa. La computacion en la nube presenta tres tipos de servicios distintos:

- Infraestructura como Servicio (IaaS): Provee almacenamiento y red para que el consumidor pueda deslplegar cualquier tipo de software
- Software como Servicio (SaaS): Permite a los clientes conectarse a las aplicaciones basadas en la nuve a traves de internet y utlizarlas.
- Plataforma como Servicio (IaaS): El consumidor controla las configuraciones del ambiente donde se encuentra su aplicacion

2.3. [Servicios web]

Un servicio web es un sistema de software diseñado para admitir la interacción interoperable de máquina a máquina a través de una red. Existen diversos estándares para compartir y consumir estos servicios, entre ellos se encuentran REST y SOAP. [1]

2.4. [Servicios web REST]

REST (Representational State Transfer): arquitectura que, haciendo uso del protocolo HTTP, proporciona una API que utiliza cada uno de sus métodos (GET, POST, PUT, DELETE, etc) para poder realizar diferentes operaciones entre la aplicación que ofrece el servicio web y el cliente.

2.5. [Servicios web SOAP]

SOAP (originalmente las siglas de Simple Object Access Protocol) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

2.6. [Java Message Service (JMS)]

Es una API que permite crear, enviar, procesar y recivir mensajes entre dos partes. Permite que un programa Java se comunique con diferentes implementaciones de servicios de mensajería de forma estándar (eliminando la necesidad de usar APIs propietarias). La comunicaciónón JMS es desacoplada y además puede ser asíncrona y/o confiable.

3. Descripción del Problema

La descripción del problema se ecuentra en el anexo "Descripción del problema".

4. Solución Propuesta

4.1. Descripción General

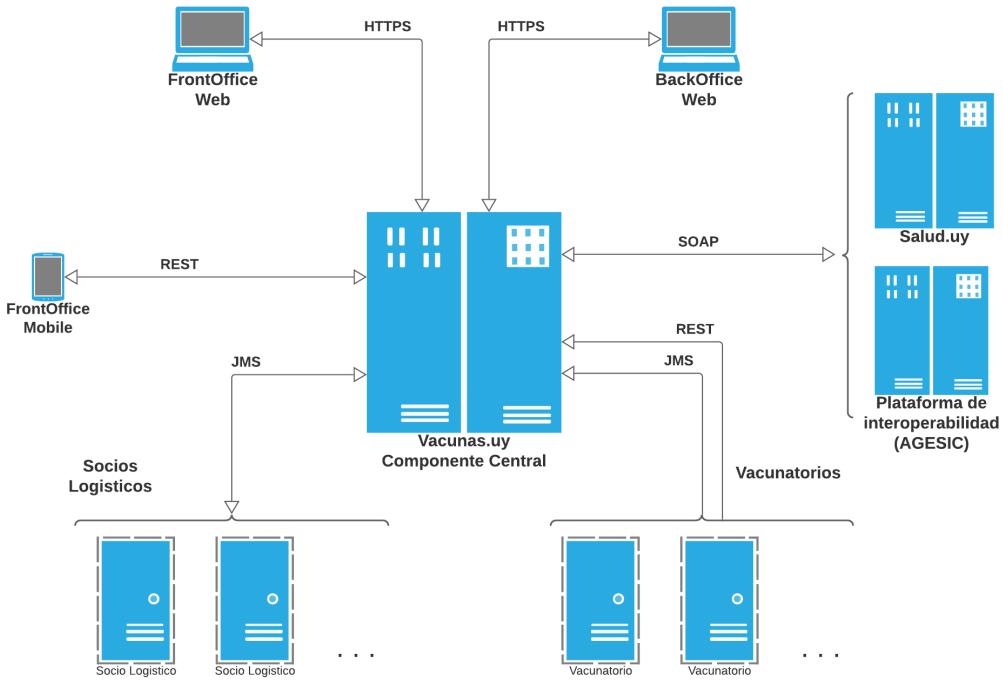

Figura 1. Descripción General de la Solucion

Para solucionar este problema se optó por la implementación de un sistema distribuido siguiendo un patrón de arquitectura en capas para los distintos componentes que se comunicarán a través de internet. Los componentes que conforman el sistema son:

- Sistema central, que contiene la lógica de negocio y brinda los servicios a los demás componentes.
- FrontOffice Móvil, que se trata de una apk que corre en dispositivos android desde donde los ciudadanos pueden consumir ciertos servicios especificos para esta plataforma.
- FrontOffice Web , sitio web enfocada a los ciudadanos y vacunadores donde podrán consumir la mayoría de los servicios pensados para 'estos.
- Backoffice Web , sitio web enfocada a autoridades y administradores para llevar a cabo sus funcionalidades administrativas en la plataforma.

- Componente para vacunatorios, es un componente de software que se comunica con el sistema central para coordinar las funciones del vacunatorio.
- Componente para socio logisticos, es un componente de software que se comunica con el sistema central para coordinar sobre el envió de lotes de vacunas.

Agenda de vacunación

Para la asignación de hora de vacunación a los ciudadano se utiliza una estrategia que permite una buena flexibilidad en cuestión a la fecha, y una flexibilidad mas moderada en cuanto al horario. El concepto agenda que manejamos en la solución se refiere a una franja de días fija que se encuentra dentro de un horario definido por un turno de un vacunatorio, que además posee la información de los intervalos de tiempo en que se divide el turno los cuales serán asignados a los ciudadanos al realizar una reserva de su vacunación, y de la capacidad de estos intervalos. Por lo tanto el ciudadano podrá elegir el vacunatorio y la fecha, pero el horario será delegado por el sistema según la elección de turno del ciudadano, y es entregado como un intervalo de tiempo en el que el ciudadano debe asistir.

Independencia entre Vacunatorios y Sistema central

Para lograr la independencia de los vacunatorios con el sistema central, en caso de que esté caído este último, se realizaron dos decisiones arquitectónicas:

- Tener una copia local de los datos relevantes al vacunatorio particular en su propio sistema, que se actualizará cuando haya conexión con el sistema central.
- Enviar los datos que sean necesarios para el sistema central a una cola de mensajes, evitando la necesidad de que el mismo esté disponible en ese momento.

Independencia entre Socio Logistico y Sistema central

Para lograr la independencia de los socios logisticos con el sistema central, en caso de que esté caído este último, se realizaron dos decisiones arquitectónicas:

- Tener una copia local de los datos relevantes al socio logistico particular en su propio sistema, que se actualizará cuando haya conexión con el sistema central.
- Enviar los datos que sean necesarios para el sistema central a una cola de mensajes, evitando la necesidad de que el mismo esté disponible en ese momento.

Chat

Se utilizó una base de datos no relacional y la tecnología provista por Firebase Real time database (Base de datos no relacional) para realizar el componente del chat. Cuando se envía un mensaje, se guarda en esta base de datos, y notifica al servidor central de un nuevo mensaje, el cual actualiza los mensajes mostrados en los navegadores usando ajax.

5. Arquitectura del Sistema

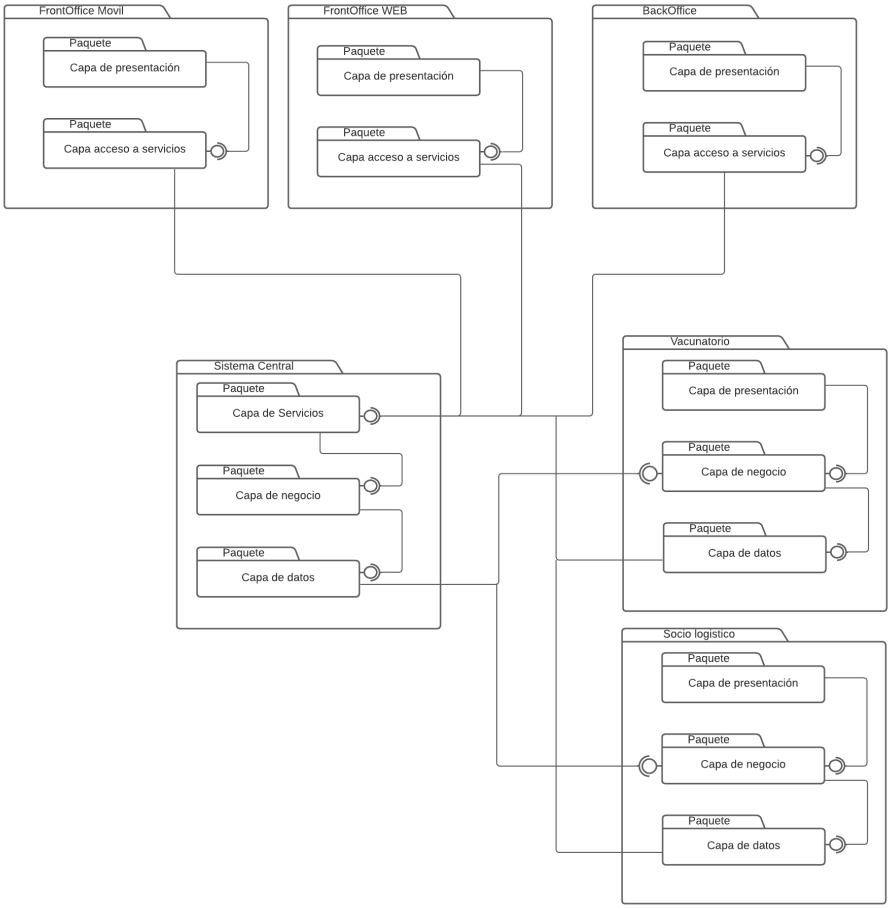

Figura 2. Diagrama de componentes

- FrotOffice Web
  Este componente se encarga de mostrar al usuario las paginas del frontoffice web destinado a ciudadanos y vacunadores y de comunicarse con el sistema central para proveer los servicios mostrados en estas paginas.
  Se divide en capa de presentación encargada de generar la vista y capa de acceso a servicios encargada de comunicarse con el sistema central.

- BackOffice Web
  Este componente se encarga de mostrar a administradores y autoridades el sitio web destinado a éstos, y de comunicarse con el servidor central para poder proveer sus servicios.
  Posee el mismo esquema de capas que el frontffice web.

- FrontOffice Móvil
  Este componente es una aplicación móvil que presenta ciertas funciones extras al frontoffice a los ciudadanos, pero mas reducida.
  Presenta el mismo esquema de capas de que el frontoffice web.

- Componente Vacunatorio
  Este componente se instala en el sistema que posean los vacunatorios y su objetivo es la coordinación de reservas de vacunación de ciudadanos, asignación de vacunadores y coordinación de entrega de dosis de vacunas, para ello se comunicará con el sistema central.
  Este componente tiene un esquema de 3 capas:
  - Capa de Presentación
    Se ocupa de la interfaz gráfica que se muestra al usuario del vacunatorio.
  - Capa de negocio
    Esta capa contiene logia de negocio necesaria para la transacción de datos con el sistema central, y para la consulta de los datos internos al vacunatorio.
  - Capa de datos
    Esta capa se encarga del acceso a los datos locales del vacunatorio.
- Componente socio logistico
  Este componente se instala en el sistema gestionado por los socios logisticos y su funcion es enviar eventos al componente central vinculados al transporte de las dosis de vacunas (p. ej. recepcion por parte del transportista, entrega de un lote de dosis a un vacunatorio, eventos generados por dispositivos IoT).
  Este componente tiene un esquema de 3 capas:
  - Capa de Presentación
    Se implementara con una interfaz grafica de comandos que se muestra al usuario del socio logistico.
  - Capa de negocio
    Esta capa contiene logia de negocio necesaria para la transacción de datos con el sistema central, y para la consulta de los datos internos al socio logistico.
  - Capa de datos
    Esta capa se tendra informacion relacionada al estado del transporte de las vacunas
- Componente central
  Este componente es el que se encuentra instalado en los servidores y que provee los servicios que consumen todos los otros componentes.
  Implementa un esquema de 3 capas:

- Capa de servicios
  Esta capa se encarga de publicar en la red los servicios que ofrece el sistema así como el empaquetado y desempaquetado de los mensajes.
- Capa de negocio
  Esta capa posee toda la lógica de negocio. Se encarga de llevar a cabo los servicios publicados por la capa de servicio.
- Capa de acceso a datos
  Esta capa de se encarga del acceso a datos del sistema central. Es responsable de pedir a la base de datos los datos que la capa de negocio necesita.

Los componentes web se implementan con el patrón MVC.

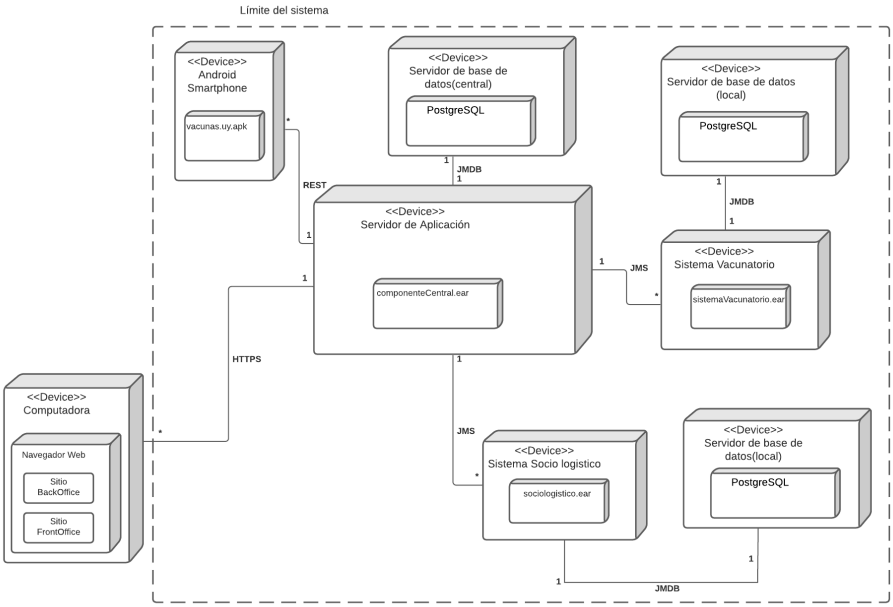

Figura 3. Diagrama de despliegue

6. Detalles de Implementación

6.1. Productos y Herramientas

Producto	Puntos Fuertes	Puntos Débiles	Evaluación General
Jakarta EE	Gran uso a lo largo de años. Resuelve problemas comunes en el mundo empresarial.	Uso directo en decadencia a favor de Spring. Cierta dificultad de acceso a documentación por este hecho.	Favorable dentro de ciertos aspectos. Sería necesario probar Spring para poder comparar de mejor manera ambas tecnologías.
JSF	Muchas librerías y herramientas. Integración fuerte con back-end Java.	Curva de aprendizaje alta. Difícil de debugear.	No tan favorable, en favor de utilizar librerías JS para el desarrollo front-end.
Firebase	Muy buena documentación. Comunidad activa. Fácil de utilizar. Ofrece muchos servicios de utilidad para facilitar el desarrollo backend.	Algunas funcionalidades son de pago a partir de cierta cantidad de usuarios. Delegar demasiados aspectos puede llevar a "vendor lock-in"	Bastante positiva, fue de gran utilidad para el manejo de las notificaciones.
PostgreSQL	Una de las tecnologías de base de datos más utilizadas, por lo tanto buena documentación y testeo. Open source. Relacional.		A efectos de nuestro sistema, ningún problema.
Maven	Útil para la gestión y construcción de software, facilita la compilación y empaquetado de código. Además permite incluir todas las dependencias necesarias para el proyecto dentro del archivo "pom"	Puede que no exista una dependencia maven para algún componente en especifico.	Positiva, nos fue de gran utilidad.
Arquillian	Framework de testing que facilita diferentes tipos de prueba en el entorno de Java EE. Acompaña los ciclos de desarrollo facilitando la creación y gestión de distintos tipos de pruebas.	Su ejecución puede ser mas lenta que otras herramientas de testing.	Nos trajo varios problemas a la hora de configurarlo, pero es la opción con mayor comunidad con diferencia.

6.2. Problemas Encontrados

Archivo de descubrimiento El documento de descubrimiento de gub.uy posee un valor para el ïssuerïncorrecto, generando problemas de validación de los tokens recibidos. Se solucionó no utilizando el documento de descubrimiento para los endpoints, y se configuraron a mano.

Arquillian Se econtraron diversos problemas para hacer funcionar los tests de arquillian. Los problemas se dieron en temas de problemas que hacían referencia a dependencias o al contenedor. Se pudieron solucionar a lo largo del tiempo, pero fueron una gran carga.

Levantar wildfly desde Eclipse
Luego de la configuración de un perfil de Maven que permitiría la ejecución de casos de prueba Arquillian surgió un problema a la hora de inicializar el servidor wildfly. El error que este mostraba era la ya existencia de los beans de la capa lógica por lo cual no lograba hacer el deploy de estos pues no pueden existir dos beans a los cuales se los referencie con el mismo nombre. La única solución que se encontró fue resintalar el IDE, en este caso, Eclipse.

Utilización de EntitityManager.merge()
Para la modificación de los usuarios del FrontOffice se vio con buenos ojos la utilización del método EntitiyManager.merge(Entity), cuya función es actualizar la entidad que comparte Id con la entidad pasada como parámetro usan con los datos de esta ultima, para lograr un cambio de roles por parte de un administrador. Pero según la estrategia que se efectúa en la implementación donde los vacunadores son una subclase de los ciudadanos este método no logró lo que se esperaba, pues identificaba que la entidad con la Id ingresada ya existía y no podía sobre escribirla, por lo cual se opto por crear una native query de tipo UPDATE, en este caso de PostgreSQL. El utilizar una query propia del manejador de base de datos propio de nuestra solución tiene una fuerte desventaja pues hace necesario traducir estas querys si es que se desea cambiar este manejador en el futuro.

Diseño de Agendar Reserva
A la hora de poner en marcha el desarrollo de la funcionalidad, surgieron varios inconvenientes no tomados en cuenta su diseño. La idea de esta funcionalidad era tener un diseño flexible y dar la posibilidad a los usuario ver los horarios disponibles y poder seleccionar uno. El primer inconveniente fue el de realizar un algoritmo, no muy complejo, para decidir cuando agendar a las siguientes dosis en caso de que la vacuna a dar así lo requiera. Para lo cual se decidió usar siempre el mismo razonamiento. Darle al usuario la misma hora y día de la semana, cumpliendo con la restricción de la mínima separación entre dosis. Esto a su vez provoca que cuando se chequea la disponibilidad de un intervalo

de tiempo determinado, hay que tener en cuenta no solo la disponibilidad de este intervalo en si mismo sino también la de los correspondientes a las dosis siguientes. Se logró realizar este chequeo de forma eficiente, para una cantidad de intervalos limitada. Se tomo la decisión de primero elegir una agenda especifica (Un determinado horario conveniente para el usuario) y luego paginar los intervalos de tiempo por semana en esos horarios. De esta forma se limita el procesamiento por cada petición y a su vez se le permite al usuario elegir un horario a su conveniencia. Luego surgió la necesidad de tener algún tipo de sincronización a la hora de efectuar la reserva, ya que para poder hacerse no se debe haber llenado el cupo de la misma. Pero se pretendía realizar un diseño no demasiado complejo ni ineficiente. Por lo que se decidió usar un atributo calculado que mantenga la cantidad de reservas hechas hasta el momento. El acceso a este atributo calculado debería poder hacerse de forma concurrente por lo cual se uso la clase AtomicInteger como wrapper. Así mismo pudiéndolo usar como semáforo a la hora de agregar una reserva a la colección.

Postgis Al intentar integrar la base de datos geoespacial a la solución, no se pudo replicar la configuración del entorno de testeo local, con el de producción. Esto dió a un error de que Hibernate no era capaz de realizar un cast a la entidad correspondiente (en el entorno de producción). Probablemente se deba a algún cambio que haya faltado replicar en el servidor, o alguna condición que hiciera falta allí y no en el entorno local, pero no se consiguió solucionar aún.

7. Evaluación de la Solución

Se apostó en la mayoría de los aspectos por un diseño con un equilibrio entre simpleza y flexibilidad. Donde se priorizo la robustez de las funcionalidades mas importante, como por ejemplo Agendar Reserva, la cual debe soportar una cantidad grande request manteniendo la performance. Además se trató de darle la mayor flexibilidad posible al usuario a la hora de elegir fecha, hora y ubicación. Por otra parte, la comunicación con los vacunatorios se hace de tal forma que los datos son replicados y almacenados en un esquema local para cada uno de ellos. De esta forma permite tener los datos necesarios para desempeñares a lo largo de la jornada con independencia, no representando un obstáculo una posible caída del sistema central. En cambio, otras funcionalidades con menor prioridad, cono es el caso del chat de vacunadores. Se decidió realizó un modelado sencillo, en el cual los mensajes tiene como destino un gran chat que los contiene todos. En caso de que la funcionalidad necesitara escalar y fuera necesario soportar otro tipo de conversaciones habría que realizar una refactorización de este modelo. Para el manejo de las sesiones se utilizaron session tokens que permiten una gran escalabilidad con respecto a manejar las sesiones a través de variables en el servidor, que tiene una fuerte ventaja frente a este ultimo, y es justamente, que no se requiere guardar estado en los servidores. Esto no solo ahorra memoria del servidor si no que también permite que físicamente el servidor que atiende a un cliente pueda variar para las distintas transacciones que este realiza, lo cual añade

también robustez al sistema pues, podría caerse el servidor que esta atendiendo a un usuario pero mientras haya otro disponible este no perderá su progreso. Se realizaron pruebas de carga usando para poder comprar la escalabilidad de la aplicación. En particular decidió probarla funcionalidad de agendar ofrecida al ciudadano. Ya que esta es la funcionalidad que mas carga debería soportar. Para estas pruebas se generó un endpoint rest que se le pasa por query params los datos necesarios para simular todos los pasos realizados al hacer una reserva. Los resultados de esta prueba se agregan en el anexo "tests".

8. Desarrollo del Proyecto

El proyecto comenzó con la elaboración de un documento de arquitectura siguiendo el esquema 4+1 de Krutchen. Una vez finalizado el documento se lleva a cabo el primer incremento con una duración de 19 días. Este incremento pretende validar la arquitectura planteada en el documento a través de un prototipo vertical para su validación con el tutor asignado. El siguiente incremento, de 19 días, tiene como objetivo un prototipo avanzado habiendo hecho las correcciones planteadas en la validación del prototipo anterior además con una mayor cantidad de funcionalidades. Finalmente se tiene un incremento de 14 días para la entrega de la versión final del sistema habiendo cumplido con todos los requisitos obligatorios.

Evaluación de los incrementos Primer incremento: Si bien su resultado fue satisfactorio no se logro desarrollar o plantear todos los aspectos arquitectónicos en este incremento (por ejemplo no se había resuelto seguridad de las sesiones de backoffice, ni tampoco el esquema de comunicación concreto a tomar entre el componente central y los vacunatorios o socios logísticos, si bien ya se sabia la tecnología). Esto se debe a que es la primera vez de todos los integrantes con estas tecnologías.

Segundo incremento: Se llegó bastante bien a esta entrega, pero debido a varios problemas con las tecnologías y herramientas, se perdió mucho tiempo en eso y no se pudo adelantar tanto como se hubiese querido, para permitir hacer todos los opcionales. Se recortará el alcance de los opcionales.

Tercer incremento: Se consiguió un resultado escalable, seguro y confiable, pero aunque se le hayan dedicado muchas horas se pudieron realizar 3 opcionales (Base de datos no relacional, encripción y sonarqube). Se llegó a implementar el opcional de bases de datos geoespacial, pero hubo muchos problemas para configurarlo en el servidor de elastic cloud, por lo que hubo que quitar esa funcionalidad.

9. Conclusiones y Trabajo a Futuro

La plataforma pudo dar una buena solución al problema, y cumple con los requerimientos de manera satisfactoria.

Trabajo a futuro

- Mejorar el chat, para permitir varias salas con los trabajadores del mismo vacunatorio en una misma sala.
- Dar más soporte al monitoreo de vacunas de parte de los socios logísticos integrando aspectos de IoT (Como por ejemplo contemplar su temperatura)
- Resolver integración con Postgis para poder tener funcionalidad de bases de datos geoespacial en el servidor.
- Realizar las demás funcionalidades opcionales.

Referencias

[1] W3 Organization, https://www.w3.org/TR/ws-arch/#whatis, 2004
[2] Taller de Sistemas Empresariales Laboratorio 2021,
https://eva.fing.edu.uy/pluginfile.php/283318/mod_resource/content/6/obligatorio_tse_2021.pdf, 2021