# ***INFORME FINAL***

*Grupo 15*
*Alonso Pablo Kevin*
*Fernandez Federico German*

## ***Introducción y presentación de la solución. Descripción del proceso diseñado.***

Elegimos trabajar con ruby ya que es un lenguaje moderno, poderoso y simple de usar que nos permitió concentrarnos en la solución del problema sin desperdiciar tiempo causado por las complicaciones que algunos lenguajes implican. Utilizamos entre otras gemas, Ruby on Rails un framework el cual simplifica y agiliza el desarrollo tanto de aplicaciones web como de APIs, devise para la autenticación y gestión de sesiones, y rolify cómo librería para el manejo de los roles.

Algunas de las tecnologías secundarias que utilizamos son PostgreSQL como motor de bases de datos debido a que además de ser una tecnología conocida por nosotros es la única que permite utilizar Heroku de manera que no solicita datos de tarjetas. JWT una nueva tecnología de seguridad para nosotros que utilizamos para brindar seguridad a nuestra API.

El proceso comienza cuando un jefe de proyecto decide realizar un nuevo proyecto, el mismo se conecta a nuestra aplicación web la cual le provee un formulario para completar los datos del nuevo proyecto, asignando los protocolos deseados, sus encargados y el orden en que se ejecutarán, entre otra información.

Una vez cargados los datos, estos son enviados al BPM por medio de la API de bonita, quien comenzará el proceso de ejecución de protocolos, seleccionando el primero a ejecutar y preparará sus datos para ejecutar.

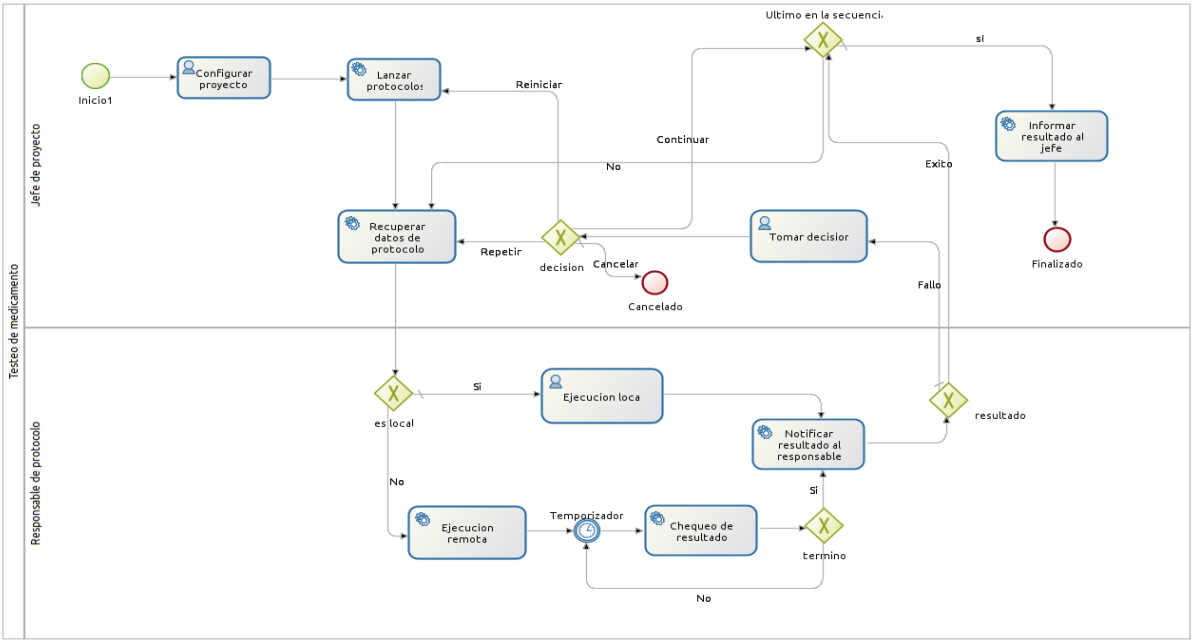
Continuando la ejecución del proceso puede tomar dos caminos dependiendo de si el protocolo previamente seleccionado se debe ejecutar localmente por el usuario o remotamente por la API. En caso de ser local nuestra aplicación web proveerá un nuevo formulario al usuario responsable del protocolo para que cargue la información necesaria de los resultados del protocolo, y en caso de que sea una ejecución remota el proceso utilizara nuestra API y le informará que debe realizar el protocolo, y luego consultará repetidamente la API cada un rango de tiempo determinado hasta que la misma termine su ejecución.

El encargado del protocolo será informado del resultado del protocolo, y en caso de falla se le informará al jefe de proyecto para que tome una de las siguientes decisiones:

- -Continuar el proceso ignorando la falla.
- -Repetir la ejecución del protocolo fallido.
- -Reiniciando el proceso, comenzando desde el primer protocolo nuevamente.
- -Cancelando y terminando la ejecución del proceso.

Y en caso de éxito continuará, evaluando si debe ejecutar más procesos, procediendo a la etapa de cargar sus datos, o si informar al jefe de proyecto sobre todo el proceso y terminando la ejecución del mismo exitosamente.

***<u>Diagrama de procesos producidos con Bonita Open Solution para el proceso de negocio</u>***




Inicio: El usuario comienza el proceso desde nuestra Web App.

Configurar proyecto: Bonita recibe por medio de su API la información del proyecto previamente cargada en el formulario de la Web App.

Lanzar protocolos: Se inicializan los valores y se selecciona qué protocolo se ejecutará inicialmente.

Recuperar datos de protocolo: Se cargan en variables de proceso la información del protocolo seleccionado.

Es local(compuerta): Se evalúa la variable previamente cargada para seleccionar el camino a tomar, si el protocolo es local o remoto.

Ejecución local: Bonita se mantiene esperando en la tarea humana hasta que recibe por medio de su API el resultado del protocolo cargado en un formulario provisto por la Web App.

Ejecución remota: Bonita se conecta con la API para loguearse en la api y enviar a ejecutar el protocolo a resolver.

Temporizador(timer): Espera un tiempo X para avanzar a la siguiente tarea.

Chequeo de resultado: Bonita se conecta con la API para recuperar el resultado (puede no estar todavía), y setea una variable indicando si ya finalizó.

Término(compuerta): Basado en la respuesta de ‘Chequeo de resultado’ vuelve al temporizador si aún no finalizó la ejecución del protocolo, de lo contrario continúa la ejecución del proceso.

Notificar resultado al responsable: Se le envía un mail al responsable con el resultado del protocolo.

Resultado(compuerta): Si el resultado del protocolo es menor al requerido falla, sino tuvo exito.

Tomar decisión: El jefe de proyecto toma una decisión en base al resultado del protocolo fallido, podrá optar por reiniciar el proceso, volviendo así a la tarea Lanzar protocolos, repetir el protocolo, volviendo a recuperar datos de protocolo, cancelar, terminando la ejecución del proceso o continuar, continuando el proceso como si hubiese sido exitosa la ejecución del protocolo.

Última en ejecución(compuerta): En base a una variable de proceso que representa cuantos protocolos restan continua la ejecución lanzando el próximo protocolo o continuando a Informar resultado al jefe.

Informar resultado al jefe: Se informa por medio de mail el resultado del proceso al Jefe de proyecto.

Finalizado: Termina la ejecución del proceso

## ***Descripción de los web services desarrollados***

Nuestra idea de la API fue que se tienen distintos protocolos cargados, pero estos no pertenecen a ningún método de testeo de medicamentos en sí, sino que desde cualquier proyecto de cualquier método de testeo que requiera ejecutar alguno de los protocolos cargados, pueden llamar a la api (en un caso real deberían pasar datos del problema, por simular la ejecución estos datos no existen, ya que no hay tareas a las cuales pasarlos). Las tareas son almacenadas como un único campo de texto, ya que al no ser algo que se ejecute realmente no encontramos el punto de que sean una clase separada que no tenga nada.

Url del servicio web: [redacted]
La duración del token es de 15 minutos
La duración de los protocolos se establece en la creación del mismo.
Para propósitos de prueba se puede indicar la duración de una ejecución en particular cuando se llama a la API con el parámetro "duración" en /execute

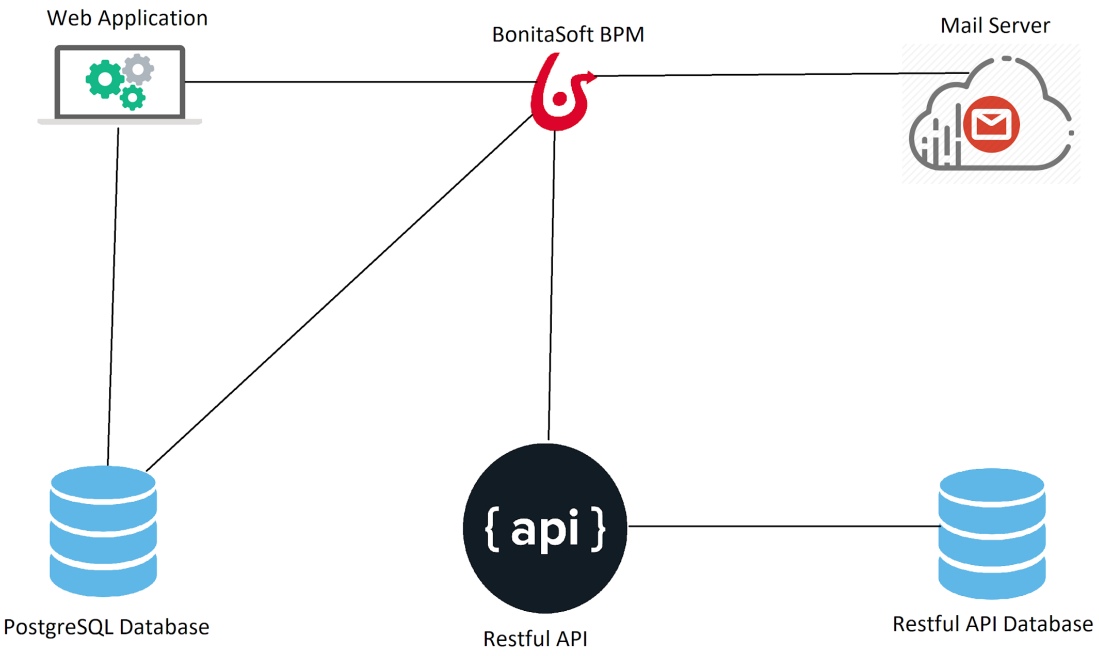
Los métodos básicos para poder realizar el ciclo de enviar a ejecutar y consultar el estado de un protocolo se encuentran resaltados.

| | | | |
|---|---|---|---|
| Login | POST | /login | Loguea el usuario con el nombre y contraseña provistos y retorna un json con el token de identificación |
| | curl -H "Content-Type: application/json" -d '{"name":"admin", "password":"1234"}' https://the-path.com/login -X POST<br>Usuarios | | |
| | | | |
| Users | POST | /users | Recibe nombre y contraseña de un usuario y lo crea (no genera el JWT, debe loguearse luego) |
| | url -H "Content-Type: application/json" -d '{"name":"admin", "password":"1234"}' https://the-path.com/users -X POST | | |
| | | | |
| | GET | /users | Retorna el JSON con todos los usuarios |
| | curl -H "Content-Type: application/json" -H "Authorization: Bearer <token>" https://the-path.com/users<br>Protocolos | | |
| | | | |
| Protocolos | GET | /protocols | Retorna el JSON con todos los protocolos |
| | curl -H "Content-Type: application/json" -H "Authorization: Bearer <token>" https://the-path.com/protocols | | |
| | | | |

| | | | |
|---|---|---|---|
| | GET | /protocols/:id | Retorna el JSON del protocolo determinado por el id |
| | curl -H "Content-Type: application/json" -H "Authorization: Bearer <token>" https://the-path.com/protocols/1 | | |
| | | | |
| | POST | /protocols | Crear protocolo |
| | curl -H "Content-Type: application/json" -H "Authorization: Bearer <token>" -d '{"nombre":"nombre de protocolo","actividades":"lista;de;actividades","duracion":"5"}' https://the-path.com/protocols -X POST | | |
| | | | |
| Estado de ejecución | GET | /status/:id | Consultar estado de la ejecución con determinado id |
| | curl -H "Content-Type: application/json" -H "Authorization: Bearer <token>" https://the-path.com/status/1 | | |
| | | | |
| Ejecutar un protocolo | POST | /execute | Enviar a ejecutar protocolo y retorna el identificador de la ejecución correspondiente |
| Sin especificar duración | curl -H "Content-Type: application/json" -H "Authorization: Bearer <token>" -d '{"protocol":1, "proyecto_id":1}' https://the-path.com/execute -X POST | | |
| Especificando duración | curl -H "Content-Type: application/json" -H "Authorization: Bearer <token>" -d '{"protocol":1, "proyecto_id":1, "duracion":3}' https://the-path.com/execute -X POST | | |

Donde <token> se reemplaza por el jwt retornado por el login

***Descripción textual y gráfica de la arquitectura completa de la solución.***




La arquitectura de nuestra solución está diseñada en base a tres programas, una aplicación web y una API ambas desarrolladas con ruby (la API siendo hosteada en Heroku), y un BPM desarrollado con BonitaSoft.

La solución comienza cuando el usuario desea crear un nuevo proyecto, para lograrlo se conecta y verifica en nuestra Web App, una vez logueado el usuario, la aplicación se conecta con la api de bonita para loguearse y obtener el token de autenticación para poder interactuar con los otros métodos de la API. Luego de completar un formulario con la información sobre el nuevo proyecto, esta información es enviada a Bonita por medio de su API, de forma que se crea una instancia del proceso y se le guardan los datos necesarios para llevar a cabo la ejecución.

Bonita se encargará de preparar los protocolos en el orden seleccionado y ejecutarlos, siendo así podría darse 1 de 2 casos en base a si el protocolo es local o remoto. Si es local, cuando el usuario responsable del protocolo ingrese a la aplicación, nuestra Web App se comunicará con Bonita para recuperar los protocolos que ese usuario tiene que completar, y luego proveerá un formulario al usuario para que complete con el resultado de la evaluación del protocolo. En caso de ser remoto Bonita se conecta a la API del servicio web, enviando los datos de autenticación y almacenando el JWT para poder identificarse, y luego envía a ejecutar el protocolo, luego se vuelve a conectar repetidas veces tras un rango de tiempo X, para recuperar el estado del protocolo, y en caso de haber finalizado, el resultado. Luego de obtener el resultado, ya sea local o remoto, se le informará al responsable del protocolo.

Dependiendo del resultado obtenido, exitoso o fallido, seguirá un camino distinto:

Fallido: Cuando el jefe del proyecto ingrese a la pantalla de inicio de la Web App se le presentaran el/los protocolos fallidos (consultados a travez de la API de Bonita), donde se le permitirá seleccionar como proceder con cada protocolo fallido, pudiendo tomar una de las siguientes medidas: reiniciar todo el proceso, re evaluar el protocolo, cancelar el proceso o continuar a pesar de la falla.

Exitoso: Bonita evaluará si aún restan procesos a ejecutar, en caso de que si se cargarán sus datos y se proseguirá con la ejecución del mismo, en caso de que no se le enviará un mail al jefe del proyecto con los resultados del proceso y termina la ejecución del mismo.

Otras relaciones entre las aplicaciones

Web App - API del servicio web: Cuando se cree un protocolo remoto en la Web App, esta se debería comunicar con la API para que también lo cree, así conseguimos mantener igualdad en las bases de datos.

BPM - Base de Datos: Bonita se encargará de conectarse a la base de datos para actualizar los datos de los protocolos remotos, y así lograr coherencia de datos, y también a fin de recuperar mails de responsables a quienes se deberán enviar informes..

BPM - Servidor de mail: Bonita se comunicara con los servidores de mail para enviar las notificaciones a los responsables de los protocolos y a los jefes de los proyectos.