1. Piezas de software

Dependiendo del alcance, si se va a hacer nada más una aplicación móvil, necesitará su desarrollo preferiblemente en tecnologías como react-native capaces de ser exportadas tanto a ios como android, consumiendo menos tiempo de desarrollo que aplicaciones dedicadas con swift o android studio. Siguiendo la idea móvil, se necesitaría una API para conectarla a la Base de datos, como base de datos usaría postgresql ya que al ser un tipo de BD transaccional, la grandes cantidades de datos y su rápido acceso nos servirán para que el Data Scientist cree modelos predictivos de ciertas funcionalidades de este prototipo como por ejemplo el tiempo de cocción de un platillo/pedido hecho entre otros. Se puede dockerizar la aplicación y buscarla desplegar en un ambiente que no necesite tantas configuraciones aunque el costo sea un poco más elevado, en servidores como heroku. Y desplegar la aplicación mediante Heroku CLI.

2. Tipo de arquitectura

Depende del alcance y tiempo, aunque en este caso se podría utilizar una arquitectura de micro-servicios, por su configuración, se suele orientar más a una arquitectura monolítica ya que es un prototipo, ya que es difícil diseñar la distribución de los microservicios, correcta, justo al inicio del proyecto y los errores en esta etapa temprana pueden ser costosos.

Lo ideal es generar la aplicación de forma monolítica y luego dividirla en los micro-servicios necesarios luego de que el prototipo cumple su función y es un prototipo exitoso.

3. Metodología

Dependiendo del alcance y el tiempo se pueden usar diferentes tipos de metodología, si bien con scrum puede salir todo muy bien, si hay el tiempo suficiente, si es requerido en un corto periodo se puede usar el Modelo de Prototipo en la ingeniería del software, el cual al estar de la mano con el feedback del cliente (en este caso el Impact Lead) se puede obtener una versión del producto en poco tiempo, la desventaja de este modelo es que si el tiempo es un factor limitante, se pueden no utilizar las mejores tecnologías para el producto final, sino las que nos permitan obtener resultados de forma más rápida y se suelen desatender aspectos importantes como la mantenibilidad y robustez del código, por lo tanto si el prototipo tiene éxito, puede verse el caso de que se tenga que reconstruir el producto de software con otra tecnología.

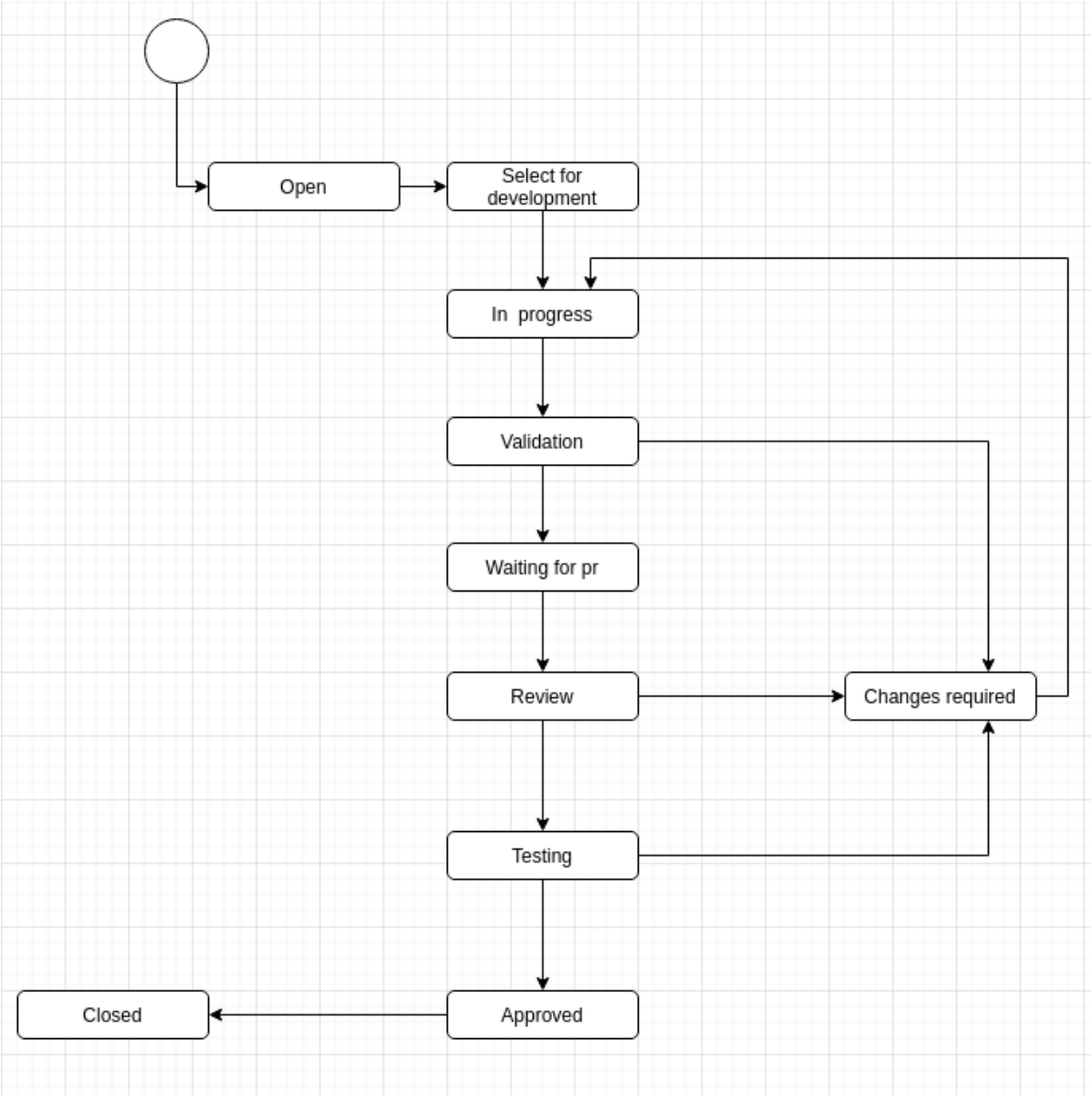
4. Workflow

El worflow a utilizar para colaborar con git, sería muy similar al de gitflow (https://danielkummer.github.io/git-flow-cheatsheet/), donde cada desarrollador identificaria su rama por el feature a realizar, de usar herramientas como jira las ramas que trabajaría el equipo seria nombre-numero_del_ticket y funcionalidad ejemplo: WEB-010/user_login y esta al terminare generaría un pull request hacia develop donde se revisará por parte del equipo de desarrollo (al menos 1 desarrollador, preferiblemente 2) antes de hacerse merge a develop

5. Consideraciones del proceso de desarrollo:

Usar alguna tecnología como jira para generar un kanban y un backlog con las tareas pendientes a desarrollar, definir un mvp y que tareas son las que cumplirian la base para el prototipo a desarrollar.

Definir un ciclo de vida para las tarjetas en las cuales diversos agentes estarán involucrados en el ciclo de desarrollo, tener un workflow de trabajo como este:



En el cual al inicio son las tareas que tenemos en el backlog, in progress es que el desarrollador empezó a trabajarla, validation sube algún video demostrando el correcto funcionamiento de la funcionalidad desarrollada, una vez validado el video, se procede a

hacer el pull request, en review se espera la revisión del pr y luego pasa al ambiente de testing a ser probada, en caso de aprobarse por qa, pasaría luego a hacerse merge del desarrollo a develop con ello se marca la tarea como cerrada.

Ricardo Cerqueira