Universidad de Antioquia
Facultad de Ingeniería

Informe final de práctica

Para obtener el título de
INGENIERO DE SISTEMAS

Realizado por
Daniel Correa Arango
Enero 15 de 2018

Título:

Análisis de datos de la plataforma SUNN para la toma de decisiones

Resumen

La plataforma SUNN presentaba una arquitectura monolítica, un mal diseño relacional en su base de datos y un desarrollo mal estructurado lo que ocasionaba constantes errores en el funcionamiento de la misma muy evidentes en la etapa de registro de los usuarios y en las conexiones entre actores. Además se tenía desconocimiento de los procesos y criterios que tomaba la aplicación para valorar los pérfiles. En base a esto se logró identificar a fondo el funcionamiento de las métricas de la plataforma, se construyó una arquitectura basada en microservicios y se inició el desarrollo de la nueva versión de la plataforma.

Introducción

La corporación Ruta N es una entidad pública que pertenece a la alcaldía de Medellín en conjunto con UNE y EPM la cual nace en el año 2009. La corporación desarrolla distintos programas y servicios para facilitar la evolución económica de la ciudad hacia negocios intensivos en ciencia, tecnología e innovación, de forma incluyente y sostenible. Ruta N busca articular y dinamizar el ecosistema de innovación de Medellín, haciendo énfasis en cuatro factores clave: la formación del talento, el acceso a capital, la generación de la infraestructura necesaria y el desarrollo de negocios innovadores. Con la aplicación de estos factores, Ruta N busca promover una cultura innovadora, la generación de empleo, el fortalecimiento de las instituciones, la formación del talento y el acceso a mercados.

Con el fin de cumplir con el principal propósito de la compañía, mejorar la calidad de vida de los habitantes, no solo de Medellín, si no de Colombia, Ruta N identificó diferentes estrategias aplicadas alrededor del mundo las cuales se presentan como una referencia tecnológica (por ejemplo, el proyecto "Distrito22"de Barcelona aplicado desde el año 2000 en dicha ciudad). Es así como la Corporación Ruta N, desde la Gerencia de Proyectos Especiales, pone en marcha el proyecto Gran Pacto MedellInnovation en septiembre de 2014 el cual, posteriormente, se convierte en el Gran Pacto por la Innovación. Este proyecto pretende incentivar la inversión de las empresas colombianas en actividades de ciencia, tecnología e innovación.

El proyecto consta de tres fases, la primera de ellas, llamada movilización y firma, consiste en incentivar a las empresas a evaluar su capacidad de innovación con una herramienta de autodiagnóstico que provee Ruta N. Al completar esta evaluación la empresa se considera firmante y puede acceder a ciertos beneficios como herramientas y asesorías para potenciar o implementar acciones de innovación. Estas acciones están contenidas dentro de la segunda fase del proyecto, denominada sistemas de innovación. Paralelamente se encuentra la tercera fase, llamada comunidad de innovación, en la cual se capacita a las empresas en el uso de las herramientas y se las introduce en una colectividad de compañías donde pueden instaurar

negocios. Una de estas herramientas de innovación es la plataforma SUNN la cual pretende conectar la oferta de innovación (grupos de investigación y startups) con su respectiva demanda (empresas e inversionistas) girando en torno a cinco ecosistemas de innovación: ciencias de la vida, materiales avanzados, energía, tecnologías limpias y tecnologías avanzadas de la información y la comunicación.

Actualmente SUNN cuenta en sus bases de datos, con gran cantidad de información de organizaciones pertenecientes a esta comunidad, entre las que se encuentran 254 startups, 361 grupos de investigación, 18 inversionistas y 1444 empresas.

La información contenida por la aplicación SUNN presenta una potencial utilidad en el proceso de toma de decisiones y del modelo de negocio en general. Sin embargo, actualmente la aplicación no presenta un diseño arquitectónico eficiente, por lo que con frecuencia se evidencian errores graves en el funcionamiento, lo que se traduce en, una pobre experiencia de usuario, un desarrollo complejo e incomprensible ya que ha pasado por manos de muchas personas sin la documentación necesaria y una escalabilidad nula ocasionada por el diseño de sus bases de datos el cual no cuenta con la integridad referencial adecuada.

Para solventar estos inconvenientes se utilizó una metodología ágil basada en SCRUM con el fin de realizar entregables periódicos, identificar errores de manera oportuna y realizar cambios en el menor tiempo posible.

Objetivos

General

Desarrollar una nueva versión de la plataforma SUNN basada en la hoja de ruta de la corporación Ruta N consolidando el modelo de negocio actualizado, aplicando buenas prácticas de desarrollo.

Específicos

- Validar la nueva idea de negocio de la plataforma con el fin de ofrecer un producto viable y atractivo de cara a los usuarios.
- Diseñar una nueva arquitectura para la plataforma, la cual sea sostenible y escalable buscando soportar un alto flujo de usuarios.
- Identificar las tecnologías más convenientes para el desarrollo de la plataforma seleccionando aquellas que se ajusten a los requisitos de la corporación
- Emplear metodologías ágiles que permitan obtener entregables funcionales en el menor tiempo posible para identificar y corregir errores.
- Crear la documentación necesaria, hoy en día inexistente, del desarrollo de la plataforma haciéndolo comprensible y mantenible para futuras intervenciones.

Marco teórico

El desarrollo de software en la actualidad se divide en dos metodologías principales, las metodologías tradicionales y las ágiles. En primer lugar, el proceso de desarrollo en una metodología tradicional se basa en técnicas predictivas, las cuales permiten estimar el alcance de un producto según sus requerimientos y el presupuesto asignado al mismo, por lo tanto todas las etapas del ciclo de vida del desarrollo deben ajustarse en gran medida al cronograma pactado al inicio del proyecto, estas etapas se describen a continuación [Stoica et al., 2013]:

- Análisis de requisitos: En este punto se analiza la necesidad del cliente con el fin de identificar la factibilidad económica, técnica y operacional del proyecto.

- Definición de requisitos: Luego de ser analizados, los requisitos deben ser debidamente documentados y aprobados por el cliente para ser diseñados y desarrollados en posteriores etapas del ciclo de vida del proyecto.

- Diseño de la arquitectura del producto: Se desarrolla una propuesta arquitectónica para el desarrollo la cual debe ser aprobada por todas las partes interesadas, en esta propuesta se deben tener en cuenta aspectos como el riesgo, robustez del producto, presupuesto y restricciones de tiempo.

- Desarrollo del producto: En esta etapa se crea el código fuente del producto basado en los requisitos y estructurado según la arquitectura definida, además se debe cumplir con especificaciones de lenguaje y framework de programación a utilizar.

- Pruebas del producto: La funcionalidad del producto debe ser probada con el fin de identificar defectos para implementar su debida solución.

- Operación y mantenimiento: Una vez el producto es probado se procede a incluirlo en ambiente de producción y liberarlo al mercado.

En metodologías tradicionales este ciclo de vida se desarrolla una única vez en todo el proyecto, entregando un producto terminado al cliente con todos los requisitos solicitados al inicio.

Por otro lado las metodologías tradicionales basan su filosofía en entregas continuas que generen valor al negocio del cliente por lo tanto todo el ciclo de vida del producto se realiza múltiples veces durante su desarrollo en los llamados Sprints con la finalidad de realizar los cambios necesarios en el momento oportuno, estos Sprints tienen una duración promedio de 2 a 4 semanas, al ser un periodo de tiempo tan corto en cada uno de ellos se desarrollan pocos requisitos del producto y se repite el ciclo de manera incremental hasta alcanzar un producto viable que pueda ser llevado a producción (Release), con el cual el cliente pueda generar retorno de inversión.

En el cuadro 1 se presenta una tabla comparativa entre metodologías de desarrollo tradicional y ágil.

Cuadro 1: Metodologías tradicionales vs Ágiles

	Tradicional	**Ágil**
Principal objetivo	Alta seguridad	Valor rápido
Modelo de desarrollo	Modelo de ciclo de vida	Modelo evolutivo
Control de calidad	Dificil y pruebas tardías	Control permanente de los requisitos y pruebas continuas
Requisitos de usuario	Detallados y definidos antes de codificar	Definición continua
Costo de reinicio	Alto	Bajo
Pruebas	Luego de que el codigo este completo	En cada iteración
Participación del cliente	Baja	Alta
Escala del proyecto	Gran escala	Todas las escalas
Requisitos	Muy estables	Propensos al cambio
Arquitectura	Ajustada a los requisitos iniciales	Diseñada para los requisitos actuales pero propensa al cambio
Costo de remodelado	Alto	Bajo

Ahora, independiente de la metodogia con la que se aborde el desarrollo del producto, es necesario establecer herramientas que faciliten el ciclo de vida del desarrollo de software, para el caso de las dos primeras etapas del ciclo (Análisis de requisitos y Definición de requisitos) existen múltiples maneras de ser abordadas, en la actualidad suele utilizarse una metodología

llamada Design Thinking ya que se incluye activamente al cliente dentro del proceso.

Design Thinkin es reconocido como un nuevo paradigma desde el año 2009 ya que se adaptó muy bien a la solución de problemas en diversos tipos de industrias, su éxito lo debe a la manera de abordar la concepción tanto del problema como su solución en donde modifica la formula que utiliza normalmente el raciocinio humano para encontrar una alternativa de solución. Normalmente el ser humano conoce muy bien el QUE se debería realizar y el COMO debe hacerse, pero tiene total desconocimiento del valor o el resultado que le otorgará este proceso, por otro lado design thinking se centra en el valor que espera tener, luego de tener muy claro este valor se procede a definir la estrategia (o el COMO) se pretende obtener el resultado esperado y por ultimo se identifica el QUE o la solución especifica para cumplir con la meta [Dorst, 2011].

Si bien design thinking permite una definición clara y precisa del problema y la estrategia a abordar, es necesario validarla, para esto es conveniente utilizar un enfoque de doble diamante, el cual consta de cuatro etapas que pretenden dar una idea prematura de la viabilidad de la solución tanto de manera técnica como de mercado [Clune and Lockrey, 2014].

El proceso de doble diamante comienza con una etapa de descubrimiento, en ésta se realiza una investigación tanto de mercado como de posibles usuarios con el objetivo de comprender si la posible solución encaja o no en algún sector del mercado haciendo el proyecto viable, en este punto es conveniente realizar encuestas o estudios de aceptación con posibles usuarios del producto. Posteriormente con la finalidad de cubrir las demás etapas del ciclo de vida del desarrollo se pasa a una etapa de definición donde se alinean las necesidades encontradas en el sector de mercado con objetivos de negocio, luego se inicia la tapa de desarrollo en la cual se crean los elementos de software necesarios, siempre teniendo presentes las pruebas de software que aseguren la calidad del producto construido y por ultimo se lleva a cabo la etapa de liberación en la cual se realiza el lanzamiento del producto en el mercado objetivo identificado en las etapas iniciales [Design-Council,].

Ahora que se puede definir de manera concisa el problema y a su vez el valor que se generará, es necesario centrarse en el modo en que será desarrollada la solución. En el desarrollo de software se identifican dos principales arquitecturas, la monolítica y la orientada a microservicios, esta primera consiste en un desarrollo centralizado donde todos los módu-

los y funcionalidades se encuentran en un mismo proyecto lo que a gran escala representa una aplicación robusta y segura, pero presenta un inconveniente al momento de abordar necesidades de escalabilidad, un ejemplo de esto es cuando se necesita escalar un modulo muy popular para los usuarios del sistema, esto implica a su vez escalar los módulos que no son tan populares lo que representa mayor consumo de recursos de los módulos que no los necesitan y, como los recursos suelen ser limitados, se limita también la capacidad de escalar. Otra característica de los sistemas monolíticos es que suelen ser desarrollados en una misma base de código con un mismo lenguaje lo que, aunque lo hace más comprensible por parte de los desarrolladores, obstruye el potencial que puede sacarse de otras herramientas, ademas, aumenta la complejidad en el caso de una posible migración [Villamizar et al., 2015].

Precisamente para solventar los inconvenientes de la arquitectura monolítica se plantea la arquitectura de microservicios, en la cual cada modulo del sistema es un proyecto a parte y por lo tanto posee sus propios datos, tecnologías, lenguajes o herramientas otorgando la posibilidad de que cada microservicio sea desplegado en un servidor a parte con propiedades específicas, así, si un modulo es mas popular para el usuario puede desplegarse en un servidor con más recursos que un servidor en el que se aloje un módulo menos popular. También, llegado el caso en el que se necesite agregar un nuevo modulo o funcionalidad solo se debe crear un nuevo microservicio y el desarrollo de este no afectara el funcionamiento de los demás. Todos estos beneficios están sujetos a un proceso de despliegue con una complejidad mas alta, el cual en la mayoría de los casos, debido a la cantidad de microservicios creados, debe ser automatizado [Jana, 2016].

En el cuadro 2 se presenta una tabla comparativa de las propiedades de la arquitectura monolítica y la arquitectura de microservicios [Lopez and Maya, 2017].

Cuadro 2: Arquitectura monolítica vs microservicios

Categoria	**Arquitectura monolítica**	**Arquitectura de microservicios**
Código	Una base de código única para toda la aplicación	Multiples bases de código. Cada microservicio tiene su propia base
Comprensibilidad	A menudo confuso y dificil de mantener	Mayor facilidad de lectura y mucho más fácil de mantener
Despliegue	Implementaciones complejas con ventanas de mantenimiento y paradas de programas	Despliegue sencillo ya que cada microservicio se puede implementar de forma individual, tiempo de inactividad minimo
Lenguaje	Normalmente desarrollado en un solo lenguaaje de programación	Cada microservicio puede desarrollarse en un lenguaje de programación diferente
Escalamiento	Requiere escalar la aplicacion entera aunque los cuellos de botella esten localizados	Puede escalarse un solo microservicio en especifico sin necesidad de alterar los demas

Metodología

Con el objetivo de desarrollar una plataforma que se ajuste al modelo de negocio planteado por Ruta N, se asistió a unas sesiones de acompañamiento con el laboratorio de creación que dispone la corporación; En estas sesiones se buscaba plantear de manera clara la necesidad que busca solventar la plataforma, identificar el mercado objetivo y sus competencias, prototipado de la solución y por ultimo aterrizar un modelo de negocio viable.

Las sesiones del laboratorio se distribuyeron en una sesión de 4 horas semanal durante cuatro semanas donde se utilizó una metodología compuesta por conceptos de design thinking (ver figura 1) y metodología doble diamante (ver figura 2).

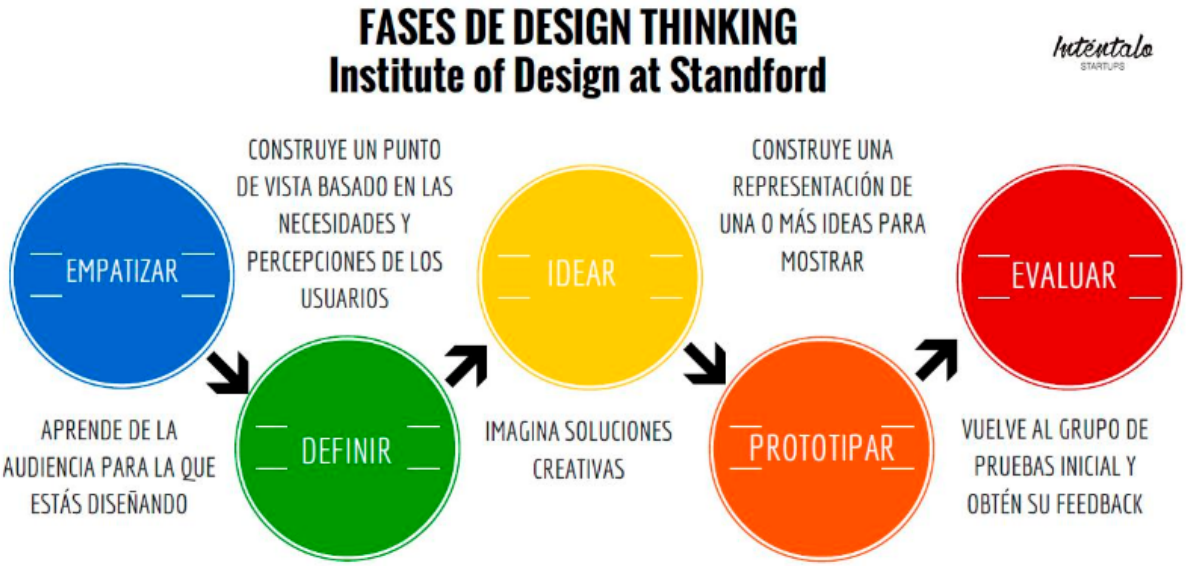

Figura 1: Proceso de la metodologia Design thinking. Tomado de `http://cursoparaemprendedoresuned.intentalo.es`

- Primera sesión: Se aterrizó la idea del negocio con el fin de que los miembros del equipo manejaran la misma terminología y se centraran en un mismo objetivo, se identificaron

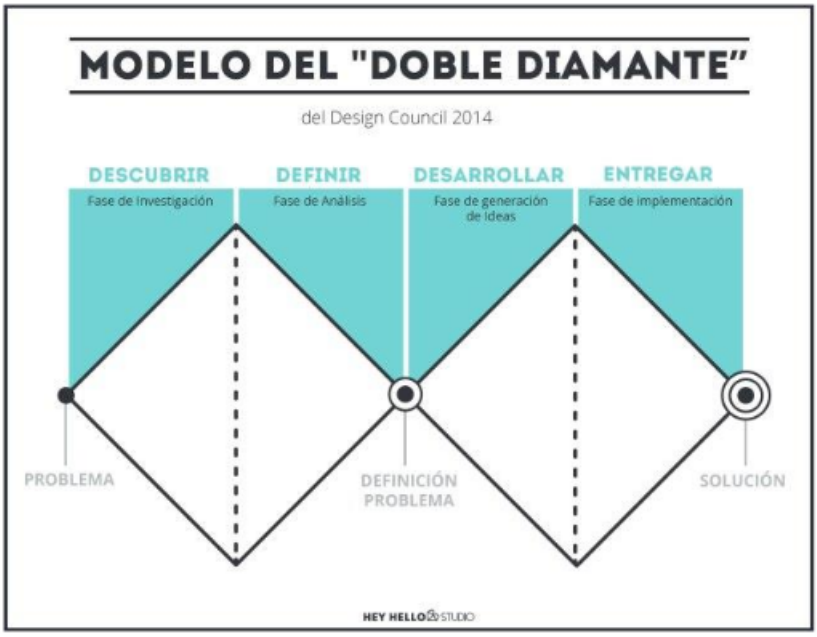

Figura 2: Proceso de la metodologia Doble diamante. Tomado de `https://i.pinimg.com`

posibles aspectos políticos , ambientales, sociales, tecnológicos, económicos y legales que pueden afectar el desarrollo o que hicieran inviable la idea de negocio, además, se detallaron los actores que interactuaran con la plataforma y su rol dentro de la misma. Esta sesión debió ser validada con un posible cliente esperando su nivel de aprobación y posibles recomendaciones o expectativas.

- Segunda sesión: Se desglosó el objetivo principal de la plataforma en momentos clave que permitieran alcanzar un producto mínimo viable, se detallaron objetivos específicos del desarrollo y se especificaron las tácticas o metodologías y los instrumentos necesarios para llevar a cabo los objetivos.

- Tercera sesión: Esta sesión se dedicó exclusivamente a la creación de prototipos de la plataforma con el fin de que fueran validados por algunos posibles clientes e identificar los puntos interesantes, ideas para incluir en futuros prototipos y críticas constructivas.

- Cuarta sesión: Esta sesión se destinó para planear el trabajo futuro y definir los pasos a seguir para cumplir con los objetivos del desarrollo, teniendo en cuenta tanto el desarrollo técnico como el de la marca, pruebas técnicas , financiamiento del proyecto, activación de la marca y mercado objetivo.

En cada una de las sesiones se llenaron formatos que permitieron sintetizar lo realizado

y guías de validación las cuales fueron diligenciadas con los posibles clientes como Tecnnova

Luego del acompañamiento con el laboratorio de creación se eligió la metodología SCRUM para el desarrollo de la plataforma gracias al constante contacto entre el equipo de desarrollo y el cliente que en este caso es la misma corporación Ruta N, así mismo, se hacía necesario un constante levantamiento de requisitos ya que solo se tenía una idea global del funcionamiento de la nueva plataforma, haber seleccionado una metodología tradicional implicaría que la totalidad de los requisitos debían ser construidos antes de comenzar el desarrollo sin oportunidad de realizar ajustes posteriores o haciéndolos técnicamente muy costosos por lo tanto la única opción viable era implementar una metodología ágil.

Se definió un primer alcance del proyecto para el primer release el cual incluía el desarrollo de los módulos de registro de usuarios y el dashboard de administración por parte del usuario broker de la plataforma, estas funcionalidades deben tener su desarrollo full stack (desde base de datos hasta la capa de visualización) el día 20 de febrero de 2018.

Al tener esta meta clara se debía seleccionar la tecnología apropiada para llevarla a cabo, se decidió utilizar una arquitectura de microservicios pues la plataforma esta concebida con un alcance en primera instancia a nivel latinoamerica pero con posibilidad de expansión a nivel global, esto implica que la plataforma debe estar abierta a incluir nuevas funcionalidades sin alterar en gran medida las ya existentes, además el alcance deja entrever que su disponibilidad debe ser muy alta, igualmente su desarrollo incremental se ve beneficiado por esta arquitectura ya que facilita la mantenibilidad del código por parte de los desarrolladores (ver figura 2).

Se utilizó la tecnologia Netflix OSS (Eureka, Feign, Zuul, Ribbon) para la arquitectura de microservicios ya que tiene una documentación bastante amplia y su uso es bajo licencia de código abierto, además se integra bastante bien con las herramientas con las que el equipo de desarrollo suele trabajar, las tecnologias utilizadas se listan a continuación:

- Eureka: Herramienta para el descubrimiento de microservicios, es una consola de monitoreo donde se observan los microservicios de la plataforma, se configuró para que constantemente revise si un microservicio dejó de funcionar y automaticamente intente recuperarlo
- Zuul: Herramienta que hace las veces de Gateway (compuerta), el cual bajo una unica ip es capaz de direccionar una petición al microservicio apropiado.

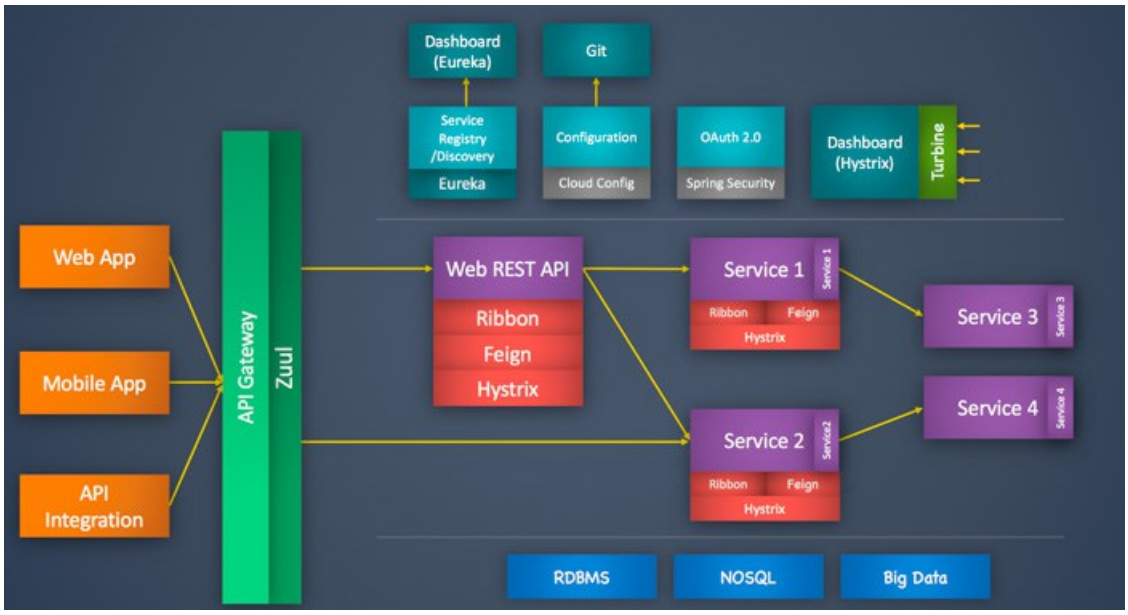

Figura 3: Arquitectura de microservicios utilizando la tecnologia Netflix OSS. Tomado de `http://www.adeveloperdiary.com`

- Feign: Herramienta para el nombramiento de servicios REST, facilita su creación y llamado
- Ribbon: Balanceador de carga en el caso de tener multiples replicas de la plataforma lo que mejora su tiempo de respuesta y solidez.
- MySQL: Motor de base de datos de código abierto, se utilizó gracias a su amplia documentación y conocimiento por parte de los desarrolladores
- Spring Boot: Framework que facilita la inyección de dependencias entre las capas de la arquitectura, es ampliamente recomendado en la documentacion de Netflix OSS
- Angular 5: Framework cuyo objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), su documentación y gran numero de funcionalidades hicieron que fuera seleccionado para el desarrollo.

Resultados

Como resultado del proceso de acompañamiento con el laboratorio de creación se crearon una serie de mockups que sirvieron para comprender a grandes rasgos el funcionamiento de la nueva plataforma, estos mockups se presentan en los anexos del 1 al 4. Ademas los formatos diligenciados de cada sesión se pueden encontrar en la siguiente dirección `https://goo.gl/ZujiKj` donde se puede observar el proceso de concepción del problema a solucionar con la nueva plataforma, las acciones necesarias a seguir y por ultimo el desarrollo y sostenibilidad de la marca.

Luego de definir la metodología de desarrollo se procedió a construir los requisitos iniciales de la plataforma los cuales se registraron utilizando la herramienta Enterprise Architect los cuales comprenden los módulos de registro de actores y de dashboard de administración (Ver figuras 4 y 5)

Estos requisitos se formularon en base a unos objetivos organizacionales que posee la corporación, por lo anterior fue necesario identificar los objetivos de negocio y presentarlos mediante una serie de casos de uso del negocio y su relación con los respectivos actores de la plataforma (ver figura 6).

En la figura 7 se describen las relaciones de cada actor del negocio con los casos de uso del negocio y la interacción con otros actores de la plataforma que para el proyecto se denominarán actores de innovación.

Si bien el Broker es un actor de innovación, en los diagramas se presenta como un actor por separado ya que puede realizar unas acciones de mayor nivel que los demás, por otro lado, también posee funciones de administración ya que coordina su propia comunidad compuesta por una cantidad definida (por una licencia) de actores de innovación.

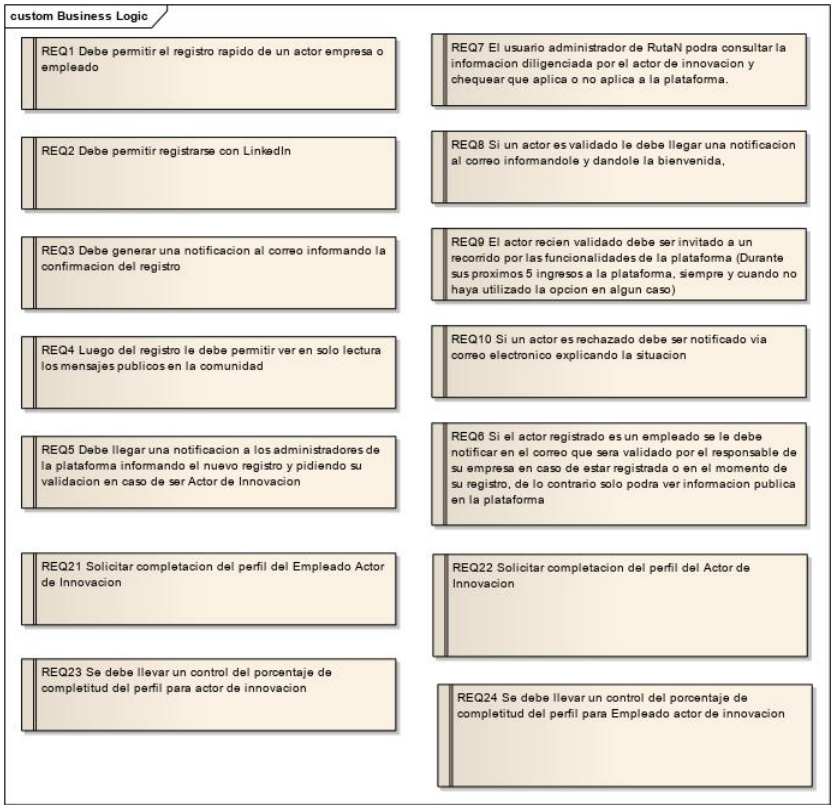

Figura 4: Historias de usuario referentes al registro de actores

Por ultimo también se diferencia el actor empleado ya que en esta nueva versión de la plataforma se permitirá el registro ilimitado de empleados de las empresas registradas, los cuales tendrán la posibilidad de compartir contenidos en una sección Timeline tipo muro, pero solo un empleado estará en la capacidad de administrar la empresa a la que pertenece, generar conexiones con los demás actores y aprobar o rechazar nuevos registros de empleados que digan pertenecer a su empresa.

Posteriormente con el fin de realizar la lógica necesaria para el dashboard de administración se realizó una tarea de estudio sobre esta misma funcionalidad en la plataforma antigua para comprender su funcionamiento, tarea que hasta ese momento no había sido realizada, al realizar el estudio se construyó un informe especificando el funcionamiento del modulo y su comportamiento que se describe por 4 métricas principales: Actividad, Atractividad, Visibilidad y Global Performance.

Cada una de las métricas se compone de uno o más parámetros, los cuales pueden ser:

- SCOUTING: Tasa de apariciones de una empresa en el radar, se calcula de la siguiente

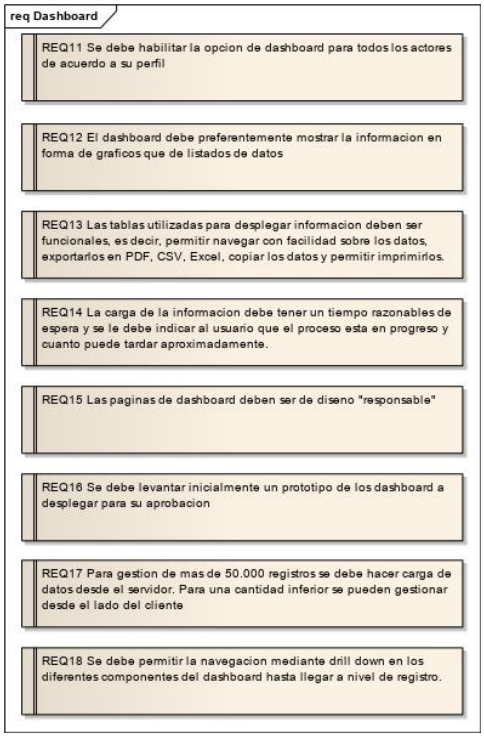

Figura 5: Historias de usuario referentes al dashboard de administración

manera

- nScouting = Numero de ocurrencias por células y región, al filtrar por célula y región se obtiene la cantidad de empresas que pueden visualizar a la compañía a la que se le quiere calcular este parámetro.
- nCompanies = Numero total de empresas.

$Valor = nScouting/nCompanies$

- PROFILE_PROGRESS: Tasa de completitud del perfil de usuario:
  - nCompleted = Numero de campos completos.
  - nTotal = Numero total de campos.

  $Valor = nCompleted/nTotal$
- PROPOSED_CHALLENGES: Numero de retos creados por la compañía.
- PROFILE_VIEWS: Numero de visitas al perfil de una empresa en el ultimo año.

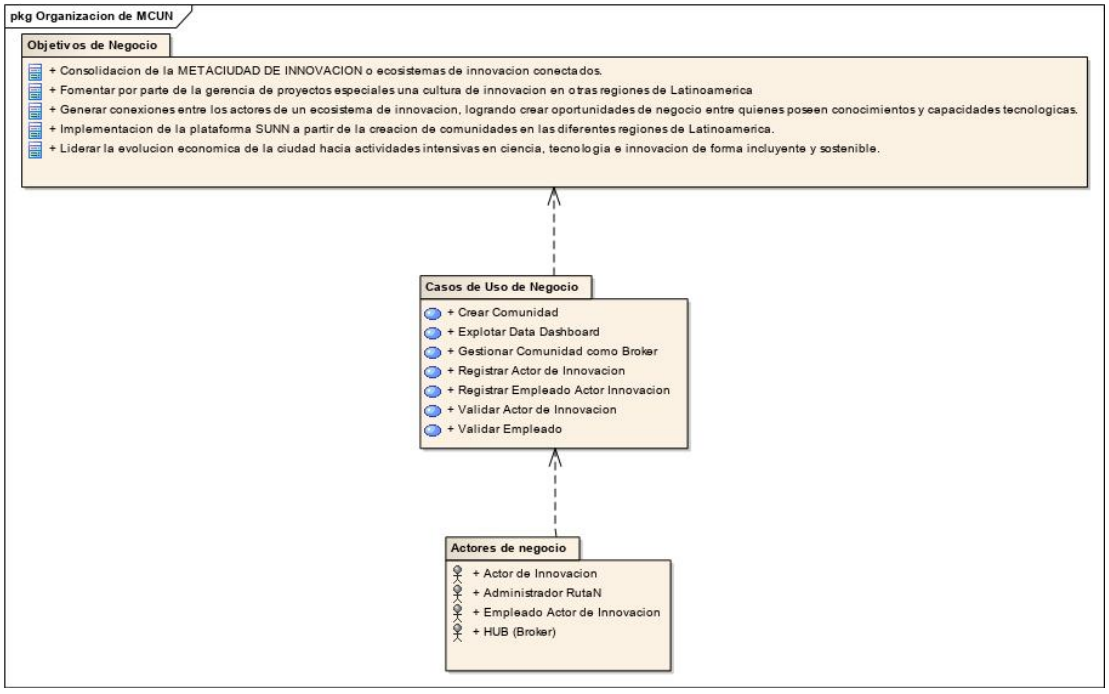

Figura 6: Relación entre los objetivos de negocio con los actores de la platafoma

- RECEIVED_WORKFLOWS: Numero de invitaciones a conectar recibidas por la compañía y aceptadas en el ultimo año.
- INITIATED_WORKFLOWS: Numero de invitaciones a conectar enviadas por la compañía y aceptadas por la otra compañía en el ultimo año.
- INNOBOX: Numero de mensajes que ha recibido la compañía en el ultimo año.

En el cuadro 3 se observan los parámetros que se utilizan para calcular cada métrica y sus correspondientes pesos y limite máximo (el limite mínimo siempre es cero). Aunque el parámetro INNOSENSOR está presente en la tabla, en la funcionalidad de la plataforma se encuentra deshabilitado, por lo tanto no aporta valor a su métrica.

Ahora para calcular el valor de cada métrica, cada uno de sus parámetros debe pasar por el siguiente proceso:

1. Se extrae el valor del parámetro como se indicó anteriormente
2. Este valor pasa por un proceso de normalización donde:

   a) si el valor es inferior a su limite mínimo, el valor se reemplaza por el limite mínimo.

Cuadro 3: Métricas del dashboard con sus parámetros

Métrica	Parámetros	Peso	Maximo
Atractividad	PROFILE_VIEWS	0.2	15
	INNOBOX	0.3	5
	RECEIVED_WORKFLOWS	1	2
	PROPOSED_CHALLENGES	0.1	5
Actividad	PROFILE_PROGRESS	0.15	1
	INITIATED_WORKFLOWS	0.8	3
	PROPOSED_CHALLENGES	0.05	5
Visibilidad	PROFILE_VIEWS	1	15
	SCOUTING	0.7	1
	INNOSENSOR	0.3	1

b) Si el valor es superior a su limite máximo, se reemplaza por el limite máximo.

c) En otro caso el valor se reemplaza por la siguiente formula:

$$valor = \frac{valor - minimo}{maximo - minimo}$$

3. Luego de obtener los valores de los parámetros, se realiza un promedio ponderado multiplicando cada valor con su respectivo peso y se divide por el peso total de la siguiente manera:

$$\frac{\sum_{n=1}^{\#parametros} valor_n * peso_n}{pesoTotal}$$

4. Por ultimo, para obtener el valor del Gobal Performance, se realiza un promedio simple entre las 3 métricas anteriores.

Para obtener las métricas de administrador, se realiza un promedio de cada métrica con respecto a la cantidad de empresas que componen la comunidad del administrador.

Luego de realizar los cálculos de las métricas, las empresas son clasificadas en varios rangos, D menos, D, D más, C menos, C, C más, B menos, B, B más, A menos, A y A más. Los valores que toma la plataforma para clasificar a las compañías son:

- Si 0 <= actividad y atractividad <0.166666667 entonces la empresa se clasifica en D menos
- Si 0.166666667 <= actividad <0.333333333 y 0 <= atractividad <0.333333333 entonces la empresa se clasifica en D

- Si 0 <= actividad <0.333333333 y 0.166666667 <= atractividad <0.333333333 entonces la empresa se clasifica en D
- Si 0.333333333 <= actividad <0.5 y 0 <= atractividad <0.5 entonces la empresa se clasifica en D mas
- Si 0 <= actividad <0.5 y 0.333333333 <= atractividad <0.5 entonces la empresa se clasifica en D mas
- Si 0.5 <= actividad <0.666666667 y 0 <= atractividad <0.166666667 entonces la empresa se clasifica en C menos
- Si 0.666666667 <= actividad <0.833333334 y 0 <= atractividad <0.333333333 entonces la empresa se clasifica en C
- Si 0.5 <= actividad <0.833333334 y 0.166666667 <= atractividad <0.333333333 entonces la empresa se clasifica en C
- Si 0.833333334 <= actividad <1 y 0 <= atractividad <0.5 entonces la empresa se clasifica en C mas
- Si 0.5 <= actividad <1 y 0.333333333 <= atractividad <0.5 entonces la empresa se clasifica en C mas
- Si 0 <= actividad <0.166666667 y 0.5 <= atractividad <0.666666667 entonces la empresa se clasifica en B menos
- Si 0.166666667 <= actividad <0.333333333 y 0.5 <= atractividad <0.833333334 entonces la empresa se clasifica en B
- Si 0 <= actividad <0.333333333 y 0.666666667 <= atractividad <0.833333334 entonces la empresa se clasifica en B
- Si 0.333333333 <= actividad <0.5 y 0.5 <= atractividad <1 entonces la empresa se clasifica en B mas
- Si 0 <= actividad <0.5 y 0.833333334 <= atractividad <1 entonces la empresa se clasifica en B mas
- Si 0.5 <= actividad <0.666666667 y 0.5 <= atractividad <0.666666667 entonces la empresa se clasifica en A menos

- Si 0.666666667 <= actividad <0.833333334 y 0.5 <= atractividad <0.833333334 entonces la empresa se clasifica en A
- Si 0.5 <= actividad <0.833333334 y 0.666666667 <= atractividad <0.833333334 entonces la empresa se clasifica en A
- Si 0.833333334 <= actividad <1 y 0.5 <= atractividad <1 entonces la empresa se clasifica en A mas
- Si 0.5 <= actividad <1 y 0.833333334 <= atractividad <1 entonces la empresa se clasifica en A mas

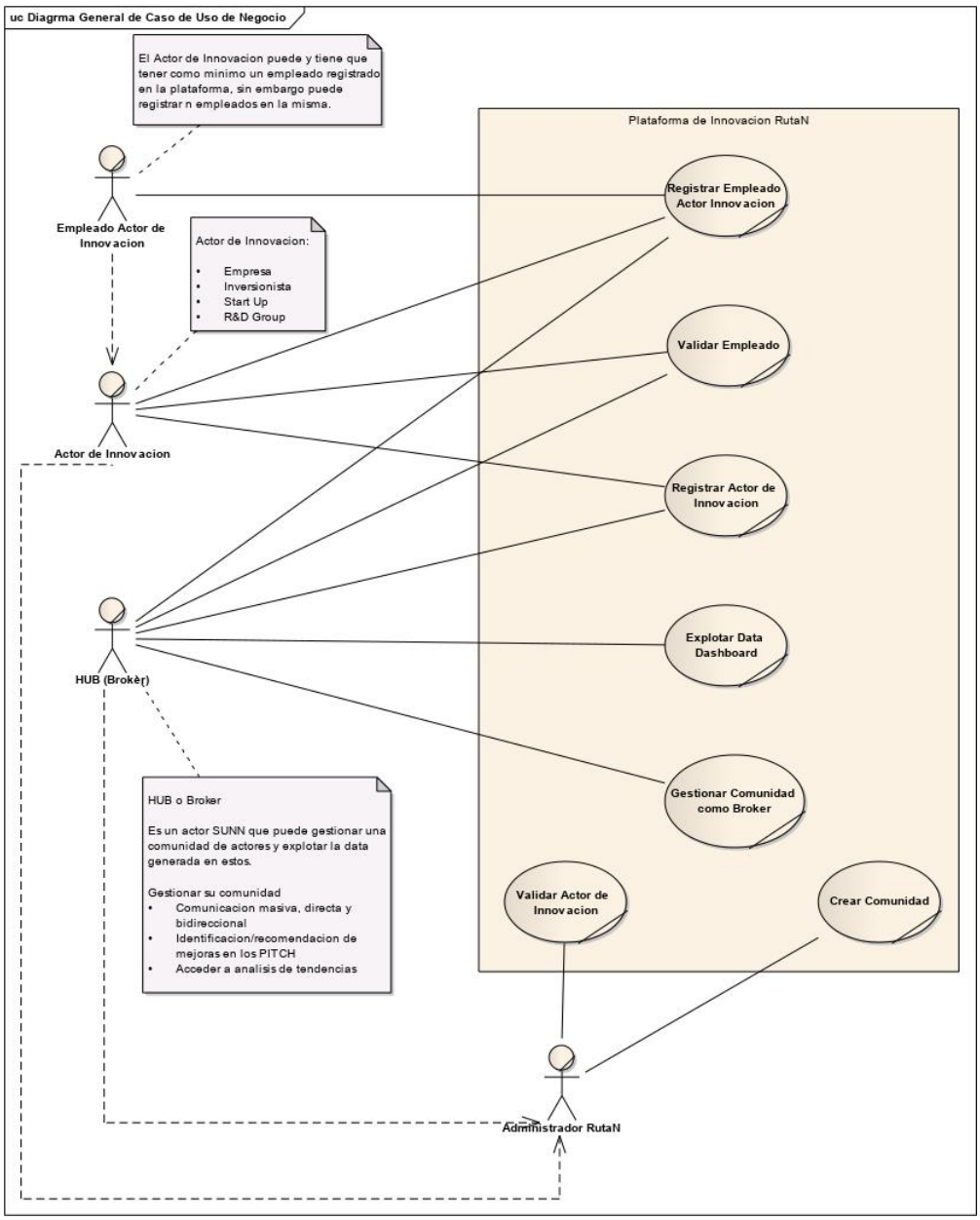

Figura 7: Diagrama de casos de uso general del negocio.

Anexos

Figura 8: Vista del landing con inicio de sesión

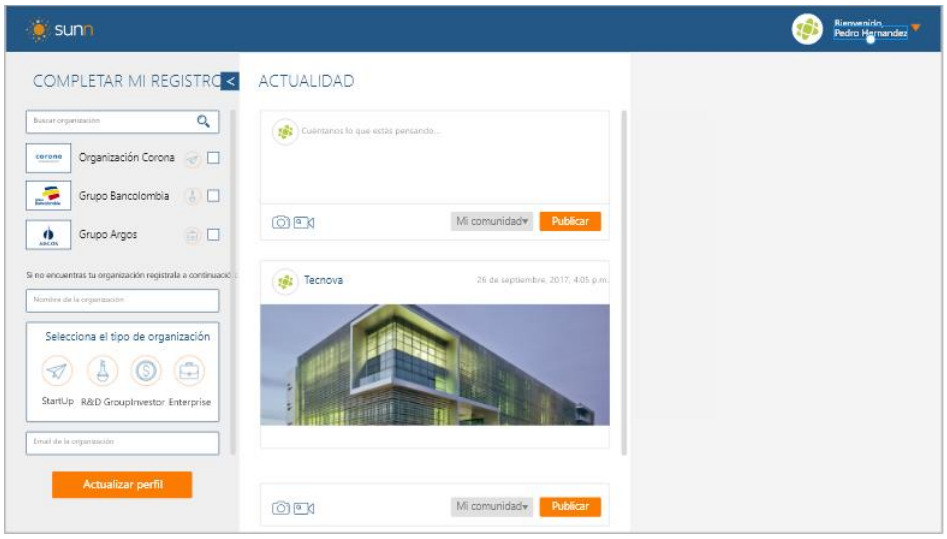

Figura 9: Sección de noticias con panel lateral para completar perfil

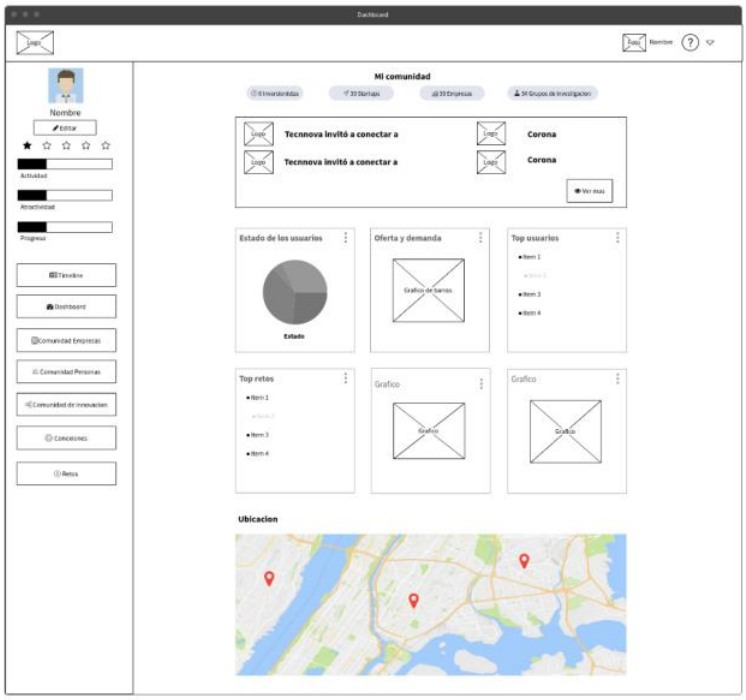

Figura 10: Métricas para administrador de comunidad

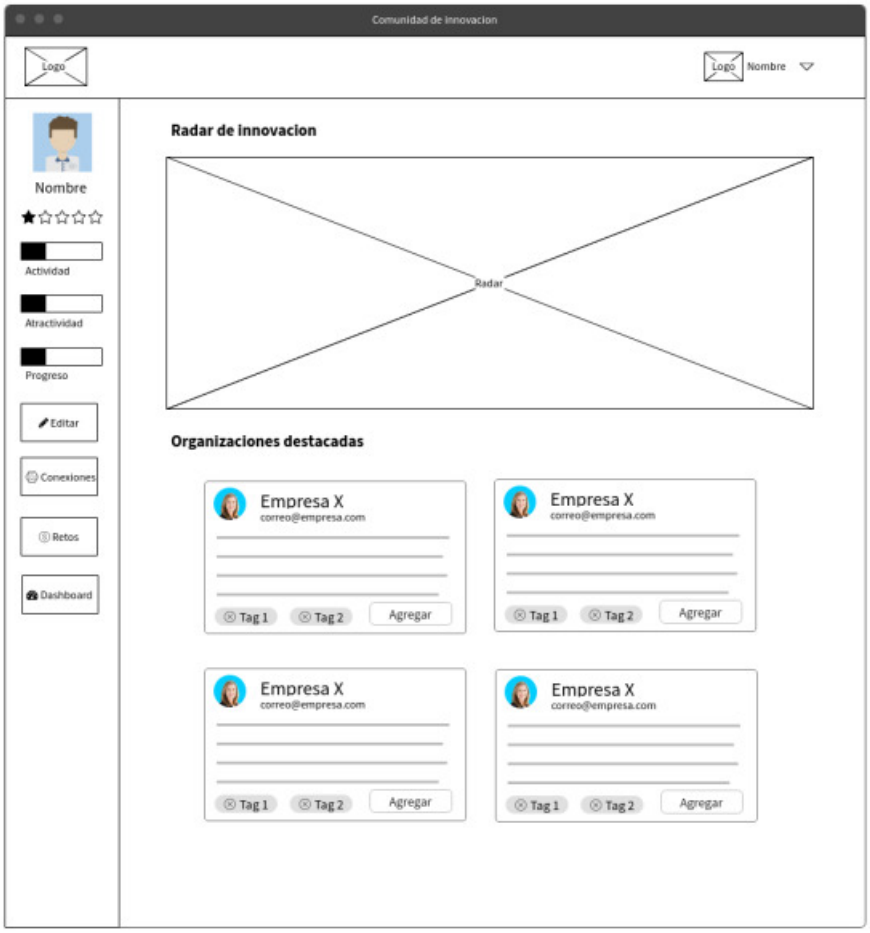

Figura 11: Sección de radar para conectar con otros actores de la plataforma

Bibliografía

[Clune and Lockrey, 2014] Clune, S. and Lockrey, S. (2014). Developing environmental sustainability strategies, the double diamond method of lca and design thinking: a case study from aged care. journal of cleaner production. 85:67–82.

[Design-Council,] Design-Council. Eleven lessons: managing designin eleven global brands a study of the design process. `https://www.designcouncil.org.uk/sites/default/files/asset/document/ElevenLessons_Design_Council%20(2).pdf`. Accessed: 2018-01-12.

[Dorst, 2011] Dorst, K. (2011). The core of 'design thinking'and its application. design studies. 32(6):521–532.

[Jana, 2016] Jana, A. (2016). Develop microservices using netflix oss and spring boot. `http://www.adeveloperdiary.com/java/spring-boot/develop-microservices-using-netflix-oss-spring-boot/`. Accessed: 2018-01-12.

[Lopez and Maya, 2017] Lopez, D. and Maya, E. (2017). Arquitectura de software basada en microservicios para desarrollo de aplicaciones web.

[Stoica et al., 2013] Stoica, M., Mircea, M., and Ghilic-Micu, B. (2013). Software development: Agile vs. traditional. informatica economica. 17(4):64.

[Villamizar et al., 2015] Villamizar, M., Garcés, O., Castro, H., Verano, M., Salamanca, L., Casallas, R., and Gil, S. (2015). Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud.