FORMULARIO B: MEMORIA TÉCNICA DEL PROYECTO	Para uso exclusivo de la ANPCyT
Título del proyecto: Parserify - Plataforma online para documentación y revisión entre pares	
Nombre del solicitante del beneficio: Sebastian Javier Marchano	

1. DIAGNÓSTICO

En este punto se debe presentar el diagnóstico que da origen al proyecto, haciendo constar, tanto las debilidades, fortalezas, amenazas y oportunidades del emprendimiento, como las posibilidades que surgen del mercado y del horizonte tecnológico.

Al redactar el diagnóstico, se debe:

a) Describir ***el problema o necesidad*** que da origen al proyecto, y la ***situación actual*** de los solicitantes en relación con el proyecto.
b) En el caso de contar con un **prototipo o versión preliminar** del producto o servicio objeto del proyecto, describir en detalle el alcance funcional, la tecnología y el estado del mismo.
c) Mencionar las principales ***características del sector productivo*** al que apunta el proyecto, con especial referencia a las características del mercado correspondiente.
d) Describir ***el potencial exportador del emprendimiento*** a partir de la situación actual del mercado extranjero y de sus requerimientos.

Si fuera necesario anexar toda la documentación pertinente que respalde o amplíe el diagnóstico.

Sobre las herramientas actuales para la compresión del software

Para que un proyecto pueda desarrollarse y lograr el objetivo debe ir acompañado de la documentación correspondiente. La información contenida en dicha documentación debe estar actualizada para que sea efectivamente el reflejo del estado del proyecto y detalle explícitamente el objetivo y propósito del mismo. Permitiendo que toda persona que se inicie en alguna de las etapas del proyecto pueda conocer y analizar cuál es la situación en la que está en ese momento.

El mundo del software no es ajeno a esto. En el proceso de desarrollo de software la necesidad y responsabilidad de tener la documentación actualizada y precisa es uno de los pasos fundamentales. Y en la actualidad esta etapa del proceso continúa siendo tan necesaria como obsoleta. A pesar de los grandes avances técnicos y tecnológicos, cuando una persona desea analizar un software debe optar por realizarlo mediante las siguientes alternativas:

1. Leer la documentación del proyecto
2. Leer el código fuente del proyecto

Nota*: Código fuente se denomina a las partes elementales que forman el software, el cual se almacena en forma de archivo de texto. El cual está escrito en lenguaje formal con el objetivo de definir, sin ningún punto de ambigüedad, unidades conceptuales de software que puedan ser transformadas mediante un compilador de lenguaje en instrucciones para ser implementadas en un ambiente de ejecución.*

La documentación que acompaña a todo proyecto de software debe ser lo suficientemente completa, precisa y actualizada. Es fundamental para el avance del proyecto que el desarrollador, que emprende un proyecto ya iniciado, se interiorice del mayor detalle posible y lo analice, para que de esta manera pueda:

- Comprender las dependencias entre otros componentes o librerías
- Comprender los algoritmos utilizados

- Estimar y comparar el consumo de recursos para cumplir el objetivo
- Conocer la interfaz de uso (APIs)
- Conocer los parámetros de configuración posibles y los límites
- Conocer, solucionar o evitar defectos de software

La documentación duplica información y es incompleta.

Actualmente existen cada vez más proyectos open source que acompañados con la gran variedad de herramientas y/o aplicaciones cuentan con mejor documentación. A esta ventaja, de contar con elementos para la correcta documentación, se debe considerar que el aporte que pueden brindar cualquiera de estas herramientas no es tan relevante si la documentación no se corresponde con el comportamiento del software que se modifica diariamente.
A pesar que es de público conocimiento la necesidad de contar con información actualizada, en muchos casos la documentación es planteada como un paso adicional a realizar, el cual demanda tiempo y esfuerzo. Esto sucede puesto que el código fuente y los datos registrados en la documentación contienen la misma información pero escritos en distinto formato. La documentación, a diferencia del código fuente, es entendible y amigable para poder llevar a cabo una comunicación más eficiente entre las personas.

Dada esta situación y con el propósito de evitar que la información sea errónea dentro de la documentación lo que se suele realizar en muchos casos es documentar parcialmente o de manera resumida, destacando los conceptos más importantes o registrando aquellos que son menos propensos a ser modificados con el transcurso del tiempo.

Por ejemplo la documentación de un proyecto podría contener:
- Diagrama de clases / objetos / paquetes
- Diagrama de secuencia / flujo / actividades
- Diagrama de despliegue / componentes
- Diagrama de interacción / casos de uso

Quedando la información referente a los detalles de implementación solamente reservados para los desarrolladores, quienes tienen la capacidad de realizar cualquier tipo de análisis sobre el código mismo del software (código fuente).

El contenido de este archivo (código fuente) está limitado con los caracteres imprimibles de la definición ASCII, American Standard Code for Information Interchange (Código Estándar Estadounidense para el Intercambio de Información) código de caracteres basado en el alfabeto latino. Por este motivo carece de formatos, conexiones y herramientas útiles para la comunicación efectiva de información como pueden ser tablas, gráficos, colores, vínculos, entre otros.

Para lograr el objetivo de la comprensión analitica del software es necesario que el interesado procese, entienda y memorice la mayor cantidad de definiciones posibles: clases, funciones, variables; para luego hacer un esfuerzo cognitivo en resumir esa información en construcciones conceptuales más precisas: interacciones de componentes, herencia de clases, invocaciones entre funciones.

El análisis que realiza la persona que se incorpora en un proyecto es:
- **Subjetivo**: el análisis de selectos componentes suelen quedar a cargo de especialistas
- **Complejo**: la complejidad radica en que un software tiene gran cantidad de componentes interrelacionados, para tener una comprensión total de una solución es necesaria la compresión total de todos los componentes
- **Temporal**: analizar el mismo problema o porción de software transcurrido un tiempo considerable puede incurrir en los mismos tiempos previamente invertidos
- **Propenso a errores**: la suposición es una herramienta común para reducir los tiempos de análisis y es la razón principal que conduce a un análisis incorrecto

- **Difícil de transmitir**: finalmente lo que se suele transmitir es sólo la conclusión del análisis

Actualmente existen herramientas pero son escasas

Para lograr mejores resultados en el análisis del código fuente se suelen utilizar IDEs (Entornos de Desarrollo Integrados, siglas en inglés). Estas herramientas no están enfocadas exclusivamente al análisis de software, tienen como objetivo principal la edición de código, compilación y ejecución del software mediante la integración de diferentes herramientas de desarrollo dentro de un mismo entorno y adaptando visualmente los componentes para brindar una mejor experiencia de usuario.
Para lograr realizar un análisis con estas herramientas lo que se suele hacer es procesar el código fuente y generar tanto perspectivas como herramientas que serán útiles en el desarrollo:

- Listado de paquetes
- Separación de código por tipo de componentes
- Listado y filtrado de variables y funciones
- Buscador de elementos por nombre
- Resúmen de dependencias
- Enlaces entre dependencias de componentes
- Integración con herramientas de control de cambios

Estas perspectivas resultan esencial para hacer un análisis eficiente.

Los principales referentes de estas tecnologías son: Eclipse, NetBeans y IntellJ. Algunas de estas cuentan con una versión web, pero la versión original se presenta en formato de aplicación de escritorio. Lo cual genera un problema subyacente con los IDEs, ya que requieren armar un espacio de trabajo, instalar software y están orientados principalmente a la edición y creación de nuevo software, no al análisis.
Los características como la accesibilidad e interacción entre personas quedan relegadas a un plano secundario.

El problema a resolver y la propuesta de valor

Como consecuencia de los problemas detallados: desactualización de los documentos, falta de registro de todas las modificaciones que se hacen y las herramientas existentes que no están diseñadas para resolver estos inconvenientes que facilitan la labor del desarrollador, y especialmente del equipo de trabajo de desarrollo, no se realizan tareas de verificación que aseguren la excelencia en la calidad del proyecto.

Es decir, existe una baja tendencia a la adaptación de prácticas de revisión de pares (peer review). Método de control de calidad que requiere que un colega de trabajo u otra persona con las habilidades necesarias pueda realizar la revisión de una modificación de código para validar los estándares del proyecto donde está incluido.

El presente proyecto pondrá foco en generar un producto que resuelva estos problemas, en particular con gran importancia en la experiencia de usuario y la respuesta del mismo para utilizar la herramienta. Esta característica permitirá una iteración veloz entre el desarrollo del producto, la puesta en producción y el análisis de aceptación de los usuarios, midiendo los comentarios en internet, la tasa de adquisición de usuarios y el tiempo de retención.

La solución propuesta

El producto es una solución pensada y desarrollada a partir de los años de experiencia en el desarrollo de software, que busca resolver las falencias anteriormente detalladas, ayudar también el trabajo del programador y el seguimiento de un proyecto de software.

Los objetivos planteados al crear esta herramienta son facilitar la comprensión y análisis de software, y la comunicación en la revisión entre pares, inicialmente para software programado en lenguaje Java y que actualmente es accesible mediante el sistema de control de versiones Git.

El servicio favorece el análisis del software brindándole al usuario información del proyecto previamente procesada. Esta etapa sólo requiere contar con la ubicación del proyecto de software como dato inicial. Desde esa ubicación y habiendo descargado el código fuente del proyecto generará las estructuras necesarias para ser utilizadas en un posterior análisis automático.

Y aporta medios para logar una eficiente comunicación para la revisión entre pares (peer review). Esto se logra mediante el acceso directo desde la herramienta por medio de notificaciones, alertando los cambios realizados por colegas para el chequeo inmediato, o mediante el envío de correo electrónico informando de la modificación efectuada, con el enlace proporcionado en el cuerpo del correo para ingresar directamente al cambio realizado.

La plataforma ha sido pensada para que pueda ser utilizada por cualquier lenguaje de software. Aún así, en un principio ha sido desarrollada para ser utilizada en todo software programado en lenguaje Java. Lo mismo sucede con el sistema de versionado, inicialmente será accesible solamente mediante el sistema de control de versiones Git, quedando luego la libertad de elegir a cada uno cuál es el sistema requerido.

La solución está ofrecida como un SaaS (software como servicio, siglas en inglés). El proyecto de software estará alojado en los servidores de un proveedor de plataforma como servicio como Digital Ocean, Google Cloud, Amazon AWS y Microsoft Azure.
Es un modelo de distribución de software donde el soporte lógico y los datos que maneja la plataforma se alojan en servidores de compañías como las mencionadas, a los que se accede vía Internet desde un cliente. La empresa proveedora de la plataforma se ocupa del servicio de mantenimiento, de la operación diaria y del soporte del software.

Beneficios

Para acceder al servicio se puede ingresar desde cualquier computadora y en cualquier momento del día mediante cualquier explorador web (Chrome, Firefox, Safari). Esto permite a los usuarios contar con los siguientes beneficios:

- Reducir los tiempos de acceso a un proyecto open source publicado en redes como Github
- Guardar y compartir análisis entre pares sobre el proyecto
- Automatizar los procesos de análisis para que se generen resúmenes, gráficos o estadísticas tomando información principalmente del código u otros análisis
- Realizar un seguimiento del resultado de dichos análisis para corroborar cómo mutan conforme el software se va modificando
- Estandarizar el aspecto visual del software, ya que ahora no es un simple archivo de texto lo que se accede sino que esta información puede estar aumentada con los análisis in situ.
- Priorizar los aspectos de accesibilidad para permitir llegar a personas con visibilidad reducida mediante las técnicas sugeridas por la WAI (Iniciativa de accesibilidad web, por las siglas en inglés) coordinado por la W3C

Características

Las características significativas del producto son:

- Poner el foco sobre la inferencia de información relevante de proyecto en función del código
- Automatización, generación de documentación y accesibilidad de la información online
- Integración con herramientas existentes de análisis estático de código
- Flexibilidad para extender la funcionalidad, accesibilidad y disponibilidad

En base a las características del producto se considera que los potenciales usuarios son los siguientes:

- **Desarrolladores**: para analizar un proyecto open source
- **Dueños de proyectos**: para hacer el seguimiento de un proyecto propio
- **Equipos**: para adoptar o mejorar las prácticas de revisión de pares

Producto preliminar

Actualmente el proyecto planteado ya cuenta con un cierto grado de avance de desarrollo y está disponible on line con una versión de prueba. Esto permite probarlo y comprobar cuáles son las funcionalidades que ofrece y especialmente evidenciar las ventajas del servicio. El enlace para acceder a la versión del proyecto es: **http://fonsoft.parserifycom**

Se adjunta el documento "Anexo - Pantallas" donde se describe en detalle el estado actual del proyecto.

Potencial exportador

El servicio ofrecido puede ser exportado a cualquier país del mundo. En primer lugar se tendrán como mercados objetivos aquellos países que contribuyan en mayor medida a proyectos open source, por ejemplo Brasil, Estados Unidos y países de Europa.

2. OBJETIVOS DEL PROYECTO

Se deben explicitar los objetivos tantos técnicos como económicos.
Este punto debe describirse, tomando como base el diagnóstico elaborado, los rasgos sobresalientes del proyecto, tales como tecnologías involucradas en los nuevos productos o servicios, impacto que tendrá en la competitividad de la empresa, etc., entre otras. En el caso de aquellos productos basados en normas o leyes, u orientados a áreas específicas por ejemplo medicina, biología, farmacia, etc., se requiere la presentación de bibliografía respaldatoria.

Los objetivos económicos están relacionados a impactos en el posicionamiento económico de los beneficiarios, como por ejemplo, apertura a nuevos mercados, profundización de mercados actuales, etc.
Este punto debe permitir comprender claramente el objetivo del proyecto

- Posicionar la plataforma como referente para el análisis automático de código y herramienta fundamental para la revisión de pares.
- Poner productivo el servicio a través de internet para penetrar rápidamente en el mercado.
- Habilitar el uso gratuito del servicio para proyectos open source como mecanismo publicitario.
- Definir qué funcionalidad y beneficio requerirá de suscripciones mensuales
- Facilitar la generación y acceso a información de calidad sobre el código fuente a toda persona que tenga permiso
- Mejorar la comunicación entre pares, ya sea entre colegas y hacia los externos al proyecto
- Disponer de diferentes configuraciones visuales y cromáticas con el propósito de mejorar la accesibilidad al código para desarrolladores con dificultades visuales.
- Otorgar al equipo fundador la posibilidad de contar con dedicación de tiempo completo para la ejecución del proyecto.

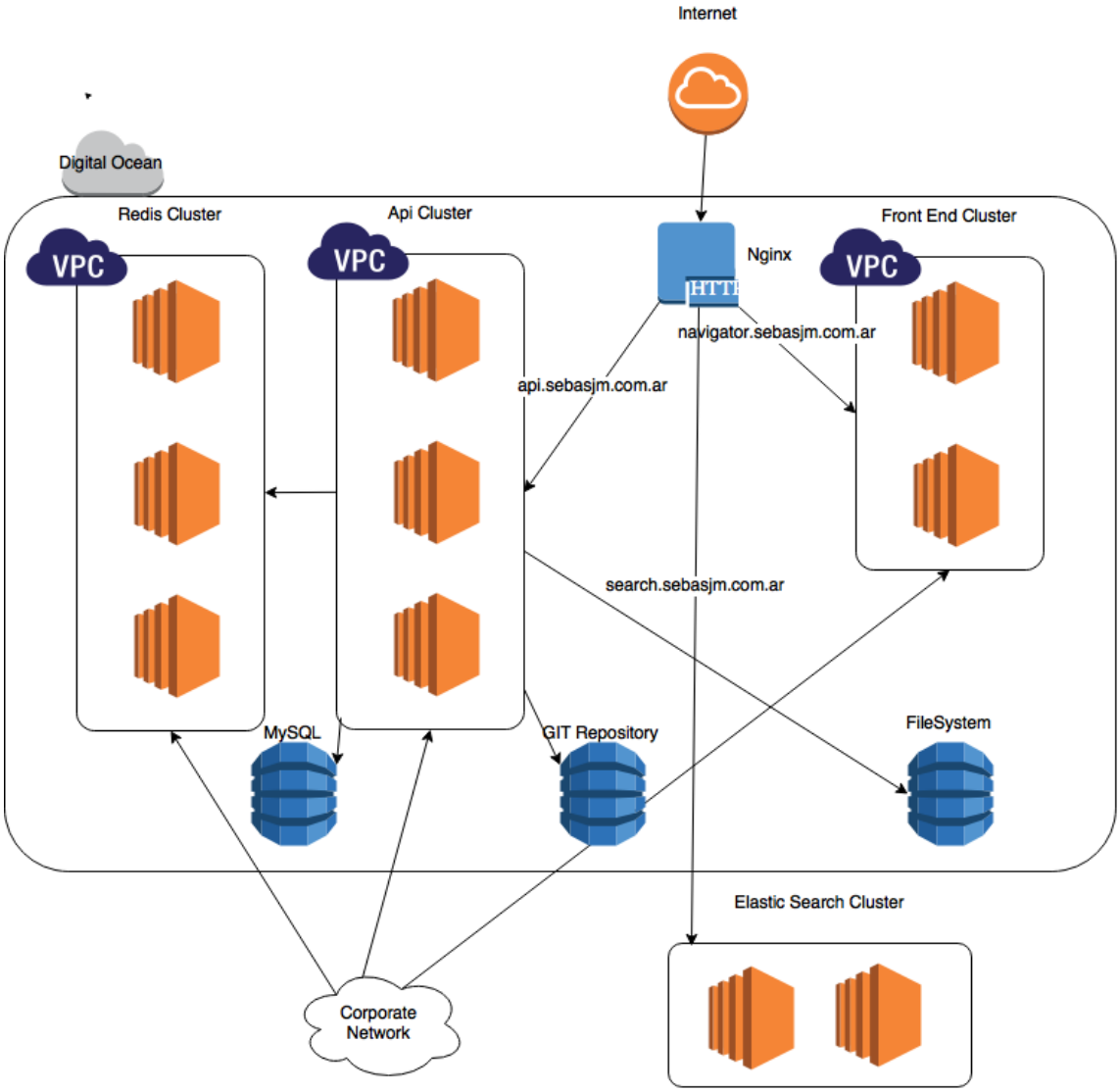
2. ALCANCE DEL PRODUCTO O SERVICIO TICS

Se debe explicitar los módulos, artefactos o piezas de software y/o hardware que se desarrollarán en el proyecto, especificando en detalle la funcionalidad de los mismos. También deberán detallarse los requisitos no funcionales.
A partir del alcance definido se deberá poder estimar el “tamaño” del producto y vincularlo con los recursos necesarios para ejecutar el proyecto con éxito, y en el plazo establecido.

Arquitectura

La arquitectura actual del sistema se divide en cinco grupos:

- WebServer Nginx
- Api Cluster
- Front End Cluster
- Redis Cluster
- Elastic Search Cluster



WebServer

Este conjunto de servers son el principal punto de acceso y direcciona los pedidos en función del dominio consultado.

El objetivo del webserver es permitir una administración de balanceo de carga inteligente y una caché que reduzca la carga a los cluster subsiguientes. En caso de alta carga, puede escalar horizontalmente ya que los mismo son independientes entre sí. El balanceo y redundancia del webserver se administra desde un ruteo de DNS.

API Cluster

Responsable de responder los requisitos funcionales del sistema, exponiendo un API Rest.
Es posible el escalamiento horizontal, ya que está desarrollado con programación orientada a eventos y servicios sin estado.
Estos servicios sin estado se comunican con una base de datos MySQL para la información que necesite ser persistida. Además, utilizan el Cluster Redis para guardar información temporal y acelerar el acceso a la misma.
Para la persistencia de los análisis, guarda la información en formato JSON (Javascript Object Notation, formato de texto ligero para el intercambio de datos) en un filesystem distribuido.

FontEnd Cluster

Encargado de asignar la aplicación web desarrollada con la tecnología ReactJS utilizando un servidor NodeJS.
La misma se distribuye en un formato SPA (Aplicación de una Única Página, por las siglas en inglés) y sólo se utiliza en la descarga de la misma para luego interactuar directamente con el API Cluster.
Cabe destacar que la aplicación es también isomórfica/universal, por lo que parte de la ejecución de rendering ocurre en el server para evitar que la misma suceda en el cliente. Logrando así un tiempo de respuesta superior y una mejor experiencia de usuario.

Redis Cluster

El mismo se usa de caché para consultas a la base de datos y para información temporal. Por ejemplo, índices necesarios para el análisis de código.

Elastic Search Cluster

El cluster de elastic search se utiliza para la funcionalidad de búsqueda, disponible desde la aplicación web. Este componente permite la indexación reversa del contenido completo del código ofreciendo tiempos de respuesta constante independientemente del total de contenido indexado.

Requisitos funcionales estimados

1. **Configuración visual de código**

   Permitirá configurar los aspectos visuales según las características del usuario, como colores y estilos de programación. Existen muchos estilos diferentes para poder adecuar el estilo de programación sin modificar el código, beneficio único de este servicio.

2. **Generación de diferencia semántica**

   En la actualidad cuando un usuario hace una revisión de un cambio la información más importante es la diferencia efectuada y esta siempre es una diferencia de linea a linea entre archivos de texto. Lo novedoso de esta solución es que la diferencia indica si el cambio contiene cambios estructurales, de dependencias, de funciones, etc..

3. **Categorización y ranking de diferencias (algorítmica, estructural, api, renombre, nulo)**

   Del punto anterior se desprende que se puede acceder a una categorización de las diferencias según la naturaleza, permitiendo un resumen general de el o los cambios.

4. **Accesibilidad para personas con dificultad visual**

Aplicando las indicaciones de la WAI (Iniciativa de Accesibilidad Web) se logra que todos estos beneficios puedan ser accesibles a un mayor público (personas daltónicas o ciegas) que hoy en día tienen mucha dificultad o es casi imposible de lograrlo.

5. **Análisis de paquetes del proyecto**
   Este análisis permite la exploración de unidades de software mediante los paquetes que lo componen.

6. **Análisis de dependencias entre clases**
   Permite la exploración entre unidades de software mediante las relaciones de dependencia como además medir la cohesión del mismo.

7. **Análisis de composición de clases**
   Este análisis posibilita la exploración entre unidades de software mediante las relaciones de composición como además medir el tamaño relativo de cada uno.

8. **Análisis de herencia de clases**
   Este análisis permite la exploración entre unidades de software mediante las relaciones de herencia, permite también conocer las diferentes implementaciones de una misma interfaz.

9. **Análisis de composición de funciones**
   Permite la exploración entre unidades de software mediante las relaciones de invocación, además en conjunto con el análisis de herencia de clase se puede hacer análisis de flujos de ejecución posible.

10. **Análisis de inferencia de tipos**
    Permite pre computar el tipo de dato del resultado obtenido para una invocación de una función.

11. **Login (Integración con auth0.com)**
    La integración de la funcionalidad de ingreso de usuario mediante el servicio de Auth0 permite una inmediata integración con diferentes redes sociales y una sencilla administración de los usuarios registrados.

12. **Timeline (Integración con getstream.io)**
    La integración de funcionalidades como notificaciones, últimas actividades, y eventos mediante el servicio de GetStream.io permite optimizar los recursos dedicados y acelerar el desarrollo de dichas funcionalidades que son transversal a la misma.

Requisitos funcionales deseables no estimados

1. Análisis de excepción en contexto
2. Análisis de bugs (Integración con FBInfer, FindBugs, ErrorProne)
3. Análisis de complejidad algorítmica
4. Análisis de cohesión de componentes
5. Análisis de configuración de proyecto (maven, gradle, makefile, buck, bazel)
6. Análisis de código duplicado
7. Detección de introducción de error
8. Code coverage in situ
9. Code performance in situ

10. Implementación de compilador para C
11. Implementación de compilador para C++
12. Implementación de compilador para Python
13. Categorización de secciones de software según acceso: Red, Disco, Mapas, Base de Datos
14. Comentarios y respuestas in situ
15. Importación de proyectos privados

Requisitos no funcionales

- **Seguridad:** la mayoría de los proyectos analizados son open source y esta información es de carácter público. Queda fuera del alcance estimado distinguir entre información pública y privada. Se utiliza conexión SSL para comunicaciones privadas, los servicios se encuentran expuestos mediante una puerta de entrada única implementada con un proxy reverso usando el motor Nginx.
- **Escalabilidad:** todos los componentes fueron pensados para ser usados concurrentemente, lo que permite escalar horizontalmente todos los servicios. Existen cuatro cluster principales web, redis, vertx y nodejs, los cuales distribuyen la carga a medida que se incrementan los nodos disponibles.
- **Disponibilidad:** al ser una solución SaaS disponible internacionalmente la accesibilidad está preparada para ser 24*7. La característica de diseño escalable permite distribuir la carga en función del uso y responder a contingencias.
- **Performance:** el SaaS se ejecuta como una aplicación SPA en el browser que se utilice. En este ambiente las mediciones de performance están relacionadas con el tiempo de respuesta visual, las cuales pueden involucrar pedidos al servidor mediante invocaciones AJAX. En total se mide que la suma de todas las interacciones no supere los 0.15 segundos.
- **Usabilidad:** estará definida por el front end, una aplicación SPA hecha con ReactJS y los últimos estándares de desarrollo web.
- **Flexibilidad:** la flexibilidad y estabilidad del servidor de API se obtiene exponiendo una interfaz, la cual se extiende conforme el sistema adquiere nueva funcionalidad. Para el caso del front end, el usuario se baja una nueva versión por cada request permitiendo una actualización de versión transparente, ambos softwares pueden ser actualizados sin interrumpir la operatoria normal durante el uso del mismo.

Si corresponde se deberá adjuntar modelos, diagramas, planos, análisis, etc. que permitan una mayor comprensión del producto o servicio TICS.
En el caso de videojuegos anexar un brief que describa como mínimo la visión del juego, la mecánica del mismo, la interfaz, los personajes, la historia, los escenarios, los elementos del juego y las especificaciones técnicas correspondientes.

3. ANTECEDENTES, ORIGINALIDAD DEL PROYECTO Y JUSTIFICACION TECNOLOGICA DEL PROYECTO

Indicar los antecedentes de la propuesta.
Indicar las características originales del producto propuesto.
Indicar las tecnologías y herramientas a utilizar para el desarrollo. Justificar la alternativa tecnológica escogida, demostrando haber considerado otras posibilidades, indicando las ventajas y desventajas de cada una.
Indicar si existen antecedentes locales e internacionales que permitan sustentar la solución tecnológica elegida en este proyecto, llevadas a cabo por el grupo técnico interviniente u otros.
Explicitar si la tecnología es de uso libre o restringido, si existen patentes directamente relacionadas con la alternativa tecnológica elegida, en el nivel nacional, y en lo posible, internacional.

La idea del producto surge a partir de los años de experiencia trabajando en el área de programación. Con el correr del tiempo y las exigencias cada vez más elevadas han hecho notar una creciente necesidad que no ha podido ser satisfecha con las herramientas disponibles hasta el momento en el ambiente de desarrollo.
Desde analizar cambios en el proyecto en el que se trabajaba, introducir nuevos desarrolladores a proyectos activos, visualizar los cambios en el contexto en el que fueron hechos, enviar las modificaciones efectuadas solamente a la gente interesada, recibir avisos sobre los cambios relevantes del proyecto y acceder a la información sin necesidad de salir del ambiente de desarrollo y desde cualquier computadora.

Para cubrir estas necesidades se han probado los siguientes repositorios de proyectos donde además de permitir alojar el código en la nube también cuentan con herramientas sociales:

- **GitLab:** es un proyecto de código libre que se puede instalar en el propio servidor y que permite tener repositorios privados, sin costo. Al ser Open Source se puede revisar el código fuente de la aplicación y hacer modificaciones. Permite generar un acceso web a los proyectos alojados en él y mediante Merge Request los desarrolladores revisarán los cambios para aplicarlos si lo creen conveniente.
- **Github:** es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git. El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago. Similar al caso anterior, se puede hacer Pull Request los que permiten solicitar cambios a proyectos de terceros.
- **BitBucket** es un servicio de alojamiento basado en web, para los proyectos que utilizan el sistema de control de versiones Mercurial y Git. Bitbucket ofrece planes comerciales y gratuitos.

También se ha probado con aplicaciones de escritorio que permiten crear y editar software. Las principales herramientas que proveen de algunos análisis similares a los propuestos son **IntelliJ**, **Eclipse** y **Netbeans**. Todas estas aplicaciones son ambientes de desarrollo integrado (IDE) para el desarrollo de programas informáticos con características similares.

Y finalmente, **UpSource** de JetBrains que es una herramienta de visualización de código, revisión de código entre pares, explorador y buscador de archivos en los repositorios configurados.

A partir del análisis de mercado buscando la mejor solución y no habiendo alguna que satisfaga todas las necesidades planteadas, es que se ha comenzado a trabajar sobre el producto que pueda cumplir con estos requerimientos.

Actualmente el proyecto planteado ya cuenta con un cierto grado de avance de desarrollo y está disponible on line con una versión de prueba. Esto permite probarlo y comprobar cuáles son las funcionalidades que ofrece y especialmente evidenciar las ventajas del servicio. El enlace para acceder a la versión del proyecto es:

http://fonsoft.parserify.com

Originalidad

La versión online sirve como producto mínimo viable y permite también comunicar las principales características que demuestran la originalidad de la implementación de la solución.

- Utilización de tecnologías web para toda la solución brindando máxima disponibilidad, accesibilidad y mínimos requisitos.
- Facilidad de integración y uso con proyectos open source disponible en plataformas como Github y Bitbucket permiten lograr una rápida adopción y viralización