 Template

2018-12-20

**About arc42**

arc42, the Template for documentation of software and system architecture.

By Dr. Gernot Starke, Dr. Peter Hruschka and contributors.

Template Revision: 7.0 EN (based on asciidoc), January 2017

© We acknowledge that this document uses material from the arc 42 architecture template, <http://www.arc42.de>. Created by Dr. Peter Hruschka & Dr. Gernot Starke.

# Introduction and Goals

Neuron Software es una herramienta encargada de centralizar la información de los diferentes proyectos que trabajan los frentes de trabajo de la organización, presentandola en un front y bajo ciertos provilegios y condiciones dependiendo del grupo de quien la necesite.

**Objetivos del Negocio**

* Aumentar en un 100% el indice de confianza en las cifras presentadas sobre el estado de las implementaciones en cada de sus etapas y a nivel general.
* Generar informes por área cada 30 días,para cada uno de los jefes.
* Consultar la información de las implementaciones en tiempo real.
* Generar un informe general cada mes,para el gerente.
* Generar un informe cada 6 meses para el área de auditoria
* Implementar durante los próximos 3 meses las vistas para cada una de las áreas;una vista por mes.

**Requisitos Funcionales**

* -El Sistema permitirá registrar diseño y/o mantenimiento nuevo.
* -El Sistema permitirá modificar diseño y/o mantenimiento.
* -El Sistema permitirá consultar en tiempo real, información de diseño y/o mantenimientos.
* -El Sistema permitirá consultar en tiempo real información de implementaciones en etapa de desarrollo y el estado de estas.
* -El Sistema permitirá modificar información correspondiente a desarrollos de implementaciones y/o mantenimientos registrados.
* -El Sistema permitirá consultar información de las implementaciones y/o mantenimientos.
* -El Sistema permitirá consultar información de las implementaciones y/o mantenimientos de sus equipos de trabajo.
* -El Sistema permitirá exportar información a PDF o EXCEL.
* El Sistema permitirá administrar la información de todas las implementaciones.
* El Sistema permitirá Generar vistas por equipo.
* El Sistema permitirá Generar vista global de todas las etapas para cada implementación y/o mantenimiento.
* El Sistema permitirá Consultar información detallada de las etapas de las implementaciones.

## Requirements Overview

## Quality Goals

|  |  |  |
| --- | --- | --- |
| Prioridad | Objetivo de Calidad | Argumento |
| Seguridad | El sistema deberá restringir el acceso ante alguna operación invalida, mediante la fragmentación de permisos y roles. | La seguridad para nosotros es importante porque se debe tener una adecuada protección de la información. |
| Usabilidad | El sistema deberá ser operado y ser utilizado con determinadas características. | La usabilidad es importante por que el sistema debe ser amigable para el usuario. |
| Mantenibilidad | El sistema debe ser tener la capacidad de ser modificado debido a necesidades | La Mantenibilidad permite que sea un software aprueba de fallas. |
| Portabilidad | El sistema deberá ser portable y ser instalado en cualquier entorno. | La portabilidad nos ayuda a que el software sea adecuado en cualquier entorno. |
| Eficiencia | El sistema deberá tener recursos adecuados que permita funcionar en condiciones determinadas. | La eficiencia nos permitirá saber el desempeño del sistema en determinadas condiciones. |
| Interoperabilidad | El sistema deberá interactuar con otros sistemas. | La interoperabilidad nos permitirá la capacidad de interactuar con los otros sistemas. |
| Fiabilidad | El sistema deberá desempeñar funciones para ser operativo y accesible cuando se requiere. | Fiabilidad nos permite tener un sistema más accesible. |
| Escalabilidad | El sistema deberá adaptarse cuando incremente su carga de trabajo. | Escalabilidad nos permitirá saber el volumen de tráfico o la ejecución de transacciones más complejas. |

## Stakeholders

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Stakeholder** | **Rol** | **Restricciones** | **Atributo de calidad** | **Viewpoints** | **Comentarios** |
| Gerente | Es la persona encargada de operar los negocios de la empresa |  | * Confiabilidad * Desempeño (Eficiencia) * Usabilidad | * Vista Contexto |  |
| Jefe de diseño | Es la persona encargada de llevar el proceso de diseño | -No puede eliminar información sin la respectiva aprobación del Gerente | * Usabilidad | * Vista Componentes |  |
| Jefe de desarrollo | Es la persona encargada de llevar el proceso de desarrollo |  | * Interoperabilidad * Mantenibilidad * Confiabilidad * Escalabilidad * Fiabilidad | * Vista Contexto * Vista Componentes * Vista Contenedores |  |
| Jefe de soporte | Es la persona encargada de llevar el proceso de soporte |  | * Mantenibilidad * Portabilidad * Escalabilidad | * Vista Contexto |  |
| Administrador del sitio | Es la persona encargada de administrar el sitio | -No puede eliminar información sin la respectiva aprobación del Gerente | * Portabilidad | * Vista Contexto |  |
| Auditoria | Es la persona encargada de auditar los procesos | -No puede eliminar información sin la respectiva aprobación del Gerente | * Seguridad * Usabilidad | * Vista Contexto |  |
| Cliente | Es la persona interesada en los servicios de la empresa | -No puede eliminar información sin la respectiva aprobación del Gerente | * Usabilidad | * Vista Contexto |  |
| Equipo de diseño | Son las persona sencargadas de llevar el proceso de diseño | -No puede eliminar información sin la respectiva  aprobación del Gerente  -Tener acceso de solo lectura a una vista con una tabla principal | * Usabilidad | * Vista Contexto |  |
| Equipo de desarrollo | Son las persona sencargadas de llevar el proceso de desarrollo | -No puede eliminar información sin la respectiva  aprobación del Gerente  -Tener acceso de solo lectura a una vista con una tabla principal | * Fiabilidad | * Vista Contexto * Vista Componentes * Vista   Contenedores |  |
| Equipo de soporte | Son las persona sencargadas de llevar el proceso de soporte | -No puede eliminar información sin la respectiva  aprobación del Gerente  -Tener acceso de solo lectura a una vista con una tabla principal | * Mantenibilidad * Portabilidad * Escalabilidad | * Vista Contexto |  |

# Architecture Constraints

Neuron Software será:

## Restricciones y lineamientos técnicas

|  |  |  |
| --- | --- | --- |
|  | **Restricción** | **Antecedente o Motivación** |
| **Restricciones de software y programación** | | |
| RT1 | Implementación en JavaScript | La aplicación debe estar construida en JavaScript, con la ultima versión del ecmascript para facilitar el performance y la latencia. |
| RT2 | Principios SOLID | Toda aplicación a implementar debe contar con la definición de SOLID dentro de su estructuración de diseño. |
| RT3 | Implementación de Pruebas Unitarias | Todo proyecto nuevo o refactorización debe contar con un modelo de pruebas unitarias, que permita garantizar la unidad mínima de código. |
| RT4 | APIs | Como un principio fundamental toda aplicación de exponer sus operaciones mediante APIs y servicios Rest para una mejor escalabilidad y mantenibilidad. |
| **Restricciones de Sistema Operativo** | | |
| RT5 | Sistema operativo | Independiente de la plataforma y debe ejecutarse en los principales sistemas operativos (Windows, Linux y Mac-OS). |
| **Restricciones de Hardware** | | |
| RT7 | Contenerización | Toda aplicación debe estar desplegada por medio de contenedores para tener un mayor control del crecimiento vertical y horizontal. |
| RT8 | Infraestructura Nube | La aplicación deberá estar desarrollada bajo infraestructura nube y por lineamientos se debe utilizar el proveedor Amazón Web Services |
| RT9 | Balanceador de Carga | Las aplicaciones que sean consideradas criticas para el negocio deberán tener configuradas un balanceador y un auto scaling. |

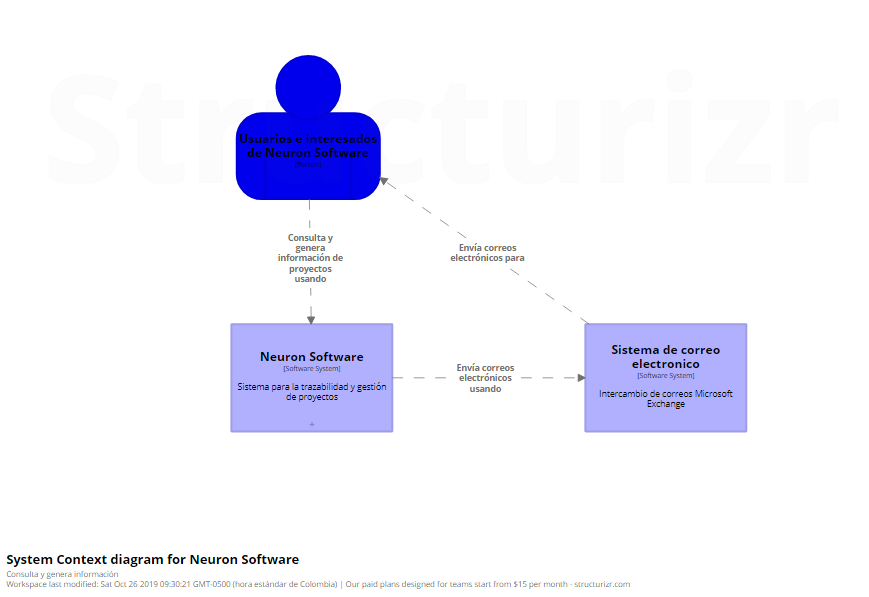
## Convenciones

|  |  |  |
| --- | --- | --- |
|  | **Restricción** | **Antecedente o Motivación** |
| C1 | Documentación de Arquitectura | Estructura basada en la ultima versión de la plantilla de Arc42 |
| C2 | Documentación de API´s y código | Se deberá utilizar suwagger tanto la plataforma como los complementos para poder realizar toda la documentación |
| C3 | Idioma | El idioma por defecto debe ser español ya que solo será utilizado a nivel de empresa. |

# System Scope and Context

Este numeral permite describir el entorno y el contexto en que el sistema va a trabajar, así mismo permitirá reconocer los componentes externos que interactuaran con el sistema.

## Business Context

****

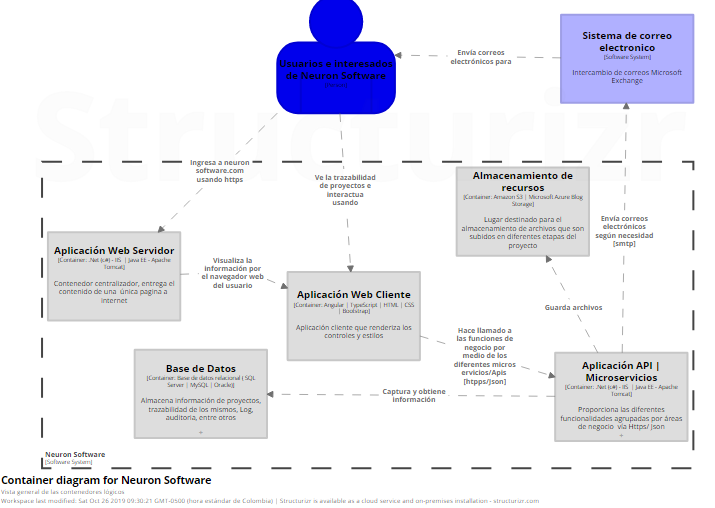
**Usuarios e interesados de Neuron Software**: los usuarios que interactuarán con el sistema se clasifican en 8 grupos, los cuales dependiendo de sus características podrán o no realizar ciertas actividades.

1. **Auditoria:** Verifican la correcta trazabilidad de la información, niveles de permisos según estándares corporativos y de entidades gubernamentales.   
   Gerente:
2. **jefe de área:** Son los encargados de planificar, gestionar y controlar las diferentes etapas del proyecto, así mismo como realizar algunas aprobaciones.
3. **Cliente**: Son los que tienen la necesidad o problema y requieren la automatización de algún proceso.
4. **Equipo de diseño:** Son las personas encargadas de tomar las necesidades, modelar y de realizar las parametrizaciones iniciales en las iniciativas y proyectos próximos a comenzar.
5. **Equipo de Implementación:** Es el equipo encargado de realizar las implementaciones y configuraciones correspondientes, partiendo de las parametrizaciones y modelado que se realice desde el diseño.
6. **Equipo de soporte:** Son las personas encargadas de proporcionar asistencia técnica cuando los usuarios presenten incidentes o algún tipo de novedad con la implementación.
7. **Gerente**: Responsable de administrar los costos del proyecto, realizar parametrizaciones generales y aprobaciones de cambio de data en las diferentes etapas.
8. **Administradores**: Son los encargados de realizar cualquier tipo de modificación ya sea a nivel de proyecto o de plataforma

**Neuron Software:** Proporciona todas las funcionalidades necesarias para satisfacer los requisitos de los diferentes usuarios e interesados como lo son parametrización de proyectos, consulta de proyectos, trazabilidad, guardar información por etapas, adjuntar archivos entre otras, que todas juntas ayudan a consolidar un gestor de proyectos.

**Sistema de correos electrónico:** Componente que proporciona todos los protocolos necesarios SMTP para el envío y recepción de correos.

## Technical Context



Neuron Software se divide en 5 contenedores o componentes principales

**Aplicación Web Servidor:** Representa el servidor de aplicaciones y todas sus características, Http, Https y sus demás protocolos donde correrá una o varias aplicaciones y APIs del sistema Neuron, de igual forma

**Aplicación Web Cliente:** Aplicación web que se ejecuta en el navegador de una maquina local, la cual renderiza todos los controles gráficos y estilos, en esta oportunidaduna aplicación web cliente centralizadora de todas las peticiones. Esta aplicación se realizará mediante el estilo SPA (Single Page Application).  
El spa se ejecuta en cualquier navegador web moderno y se comunica a través de http con el api

**Bases de Datos:** Este contenedor representa los servidores de base de datos que se utilizarán, los cuales pueden llegar a ser uno o varios, que a la vez tendrán toda la configuración y seguridad necesariapara un correcto funcionamiento. Se tendrán 5 bases de datos distribuidas en los diferentes servidores, apuntando cada una a su correspondiente api.

**Aplicación Api | Microservicios:** Este contenedor dedicado al funcionamiento de las Apis en su interior se dividen en 5 componentes, donde cada uno tiene una función y una responsabilidad especifica [ApiProyecto, ApiLoginPermisos, ApiEmail, ApiCliente, ApiAuditoria]. Estas Apis se ejecutan en servidores de aplicaciones compatibles y multiplataformas, utilizando contenedores integrados para una escalabilidad mucho más optima. Estas Apis expondrían métodos públicos que se consumirían desde un FrontEnd centralizador, algunas de estas Apis se comunicarán entre ellas, como es el caso del ApiAuditoria, que será instanciada desde las demás.

**Almacenamiento de Recursos:**Sera un contenedor de almacenamientode archivos, donde la aplicación podrá guardar físicamente todos los adjuntos cargados por los usuarios, contando con una segregación por proyectos, los nombres de los archivos serán guardados en una tabla junto con el nombre de la subcarpeta del proyecto, para armar lógicamente la ruta de cada archivo.  
la manipulación del almacenamiento será por medio del Api expuesto por el proveedor de nube para este servicio

Algunas de las interfaces principales y el canal de comunicación las listamos a continuación

|  |  |
| --- | --- |
| Interfaces de negocio | Canal |
| Login | ApiLoginPermisos (Http) |
| Registro de clientes | ApiClientes(Http) |
| Registro de Proyecto | ApiProyecto(Http) |
| Registrar Diseño | ApiProyecto(Http) |
| Registrar Implementación | ApiProyecto(Http) |
| Envió de correos | ApiEmail(Http) |
| Guardar Errores | ApiAuditoria(Http) |
|  |  |

# Solution Strategy

Neuron Software, es un aplicativo que permite llevar toda la trazabilidad de iniciativas y proyectos en sus diferentes etapas.

El corazón de Neuron software es un modelo de dominio simple manejado por microservicios donde se dividen en 5 grandes grupos [ Proyectos, clientes, LoginPermisos, Auditoria,Email].

de esos 5 grupos principales, el microservicio proyecto es el que agrupará todas las funcionalidades necesarias para llevar a cabo toda la trazabilidad de los proyectos, contando con tres entidades fuertes que son Diseño, Implementación, soporte.

La aplicación utilizará Repositorio para el manejo de persistencia de datos, de igual forma toda la lógica que se requiera en cuanto a sentencias SQL se encapsularán en procedimientos almacenados, permitiendo con esto una mayor seguridad en el manejo de la data y reutilización.

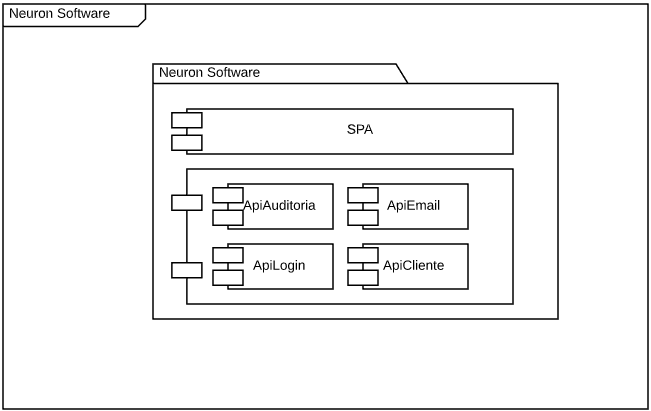
El FrontEnd se construirá mediante el estilo SPA (Single Page Application) lo cual ayuda a una mejor experiencia de usuario disminuyendo considerablemente la latencia entre opciones, ya que con SPA el contenido se reescribe dinámicamente en una misma página, que es cargada con toda la información desde el inicio, apoyados de tecnologías tipo cliente como Angular, JavaScript, Json y la apariencia visual que proporciona Bootstrap.

Desde la parte del BackEnd estará construida modularmente, permitiendo mayor escalabilidad y control entre las diferentes apis y microservicios.

La interoperabilidad se logrará mediante el uso de JSON sobre el protocolo http simple para los microservicios mencionados anteriormente, La seguridad no es el atributo principal del proyecto por lo cual debe ser lo suficientemente seguro como para evitar que otros se mezclen con los datos de otros roles. El atributo principal con el que se debe enfocar el proyecto es el rendimiento, tener un buen performance en cada una de las peticiones es prioridad para lograr esto es necesario implementar esquemas de optimización de consultas, índices que permitan una mayor velocidad y eficiencia en las diferentes consultas al mismo tiempo que se implementara

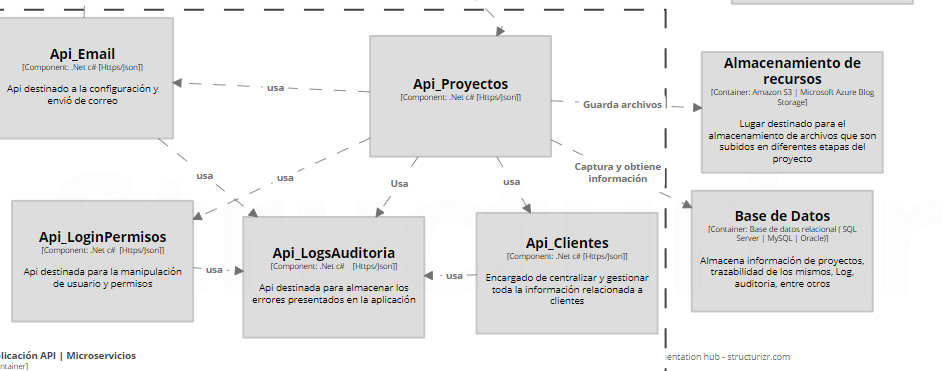
# Building Block View

La aplicación Neuron Software tiene varios empaquetados, que contienen sus principales componentes



De estos elementos se hará mayor énfasis a los componentes del Api detallando un poco sus elementos internos

## Whitebox Overall System



Partiendo del diagrama de contenedores o de contexto técnico, se realiza un zoom al contenedor de Apis, donde en su interior vemos cinco Apis totalmente autónomas e independientes que tienen diferentes responsabilidades. Muchas Apis se comunican entre ellas o requieren compartir objetos, para no tener una dependencia entre las Apis y tener un bajo acoplamiento, la comunicación entre ellas se realizará por medio de eventos que disparará la acción en el respectivo.

### White Box *Neuron Software: ApiProyecto*

El siguiente diagrama muestra los principales bloques de construcción del ApiProyecto y como estos se relacionan.



Los seis componentes principales que conforman el ApiProyecto tienen responsabilidades diferentes que se encapsulan dentro del microservicio

|  |  |
| --- | --- |
| Componente | Descripción |
| Servicio | Este componente interno sera el encargado de centralizar todas las funciones y reglas de negocio que se utilicen en el Api, adicionalmente tendra segregación por opción. |
| IServicio | Componente interno encargado de exponer las firma o contratos de los metodos y funciones realizados en la capa servicios a las otras capas |
| Repositorio | Capa encargada de la manipulación y persistencia de datos |
| IRepositorio | Componente interno encargado de exponer las firma o contratos de los metodos y funciones realizados en la capa DAO – Repositorio para ser consumidos por la capa de servicios |
| Domino | Representación de objetos y tablas que se utilizan para la correcta manipulación de los datos |
| Entidades | Representa los objetos de las clases y dominios que se utilizaran en diferentes etapas y capas |

Todos los componentes anteriores son similares para todas las Apis(Microservicios) lo que cambia es el dominio de negocio y el conjunto de operaciones

***Interfaces***

En la siguiente tabla se describen las principales interfaces que expone el ApIProyecto

|  |  |
| --- | --- |
| Interfaz | Descripción |
| /api/Neuron | Interfaz (API RestFull) encargada de las operaciones fundamentales (GET, POST, DELETE y UPDATE) de los proyectos que manejará la organización. |

### ApiProyecto : Api/diseno

***Responsabilidad***

Esta interfaz es la encargada de realizar toda la gestión e interacción necesaria cuando los proyectos o iniciativas esten en la etapa de diseño, esto involucra las operaciones principales del http.

***Interfaces – Métodos***

|  |  |
| --- | --- |
| Interfaz | Descripción |
| /api/diseno/ | Consulta todos los proyectos en la etapa de diseño, mediante el protocol [HttpGet] |
| /api/diseno/{id} | Consulta un Proyecto especifico en la etapa de diseño, mediante el Código de identificación del Proyecto y utilizando el protocolo [HttpGet] |
| /api/diseno/setRegistrar/ | Encargado de registrar toda la información y adjuntos en la etapa de diseño por medio del protocolo [HttpPost] |
| /api/diseno/setModificar/ | Encargado de modificar toda la información y adjuntos en la etapa de diseño por medio del protocolo [HttpPost] |

***Archivos***

Los archivos y elementos que se adjunten tanto en Registrar como en Modificar serán almacenados en el recurso S3 de Amazon Web Service.

### ApiProyecto : Api/Implementación

***Responsabilidad***

Esta interfaz es la encargada de realizar toda la gestión e interacción necesaria cuando los proyectos o iniciativas esten en la etapa de implementación, esto involucra las operaciones principales del http.

***Interfaces – Métodos***

|  |  |
| --- | --- |
| Interfaz | Descripción |
| /api/implementacion/ | Consulta todos los proyectos en la etapa de implementación, mediante el protocol [HttpGet] |
| /api/ implementacion /{id} | Consulta un Proyecto especifico en la etapa de implementación, mediante el Código de identificación del Proyecto y utilizando el protocolo [HttpGet] |
| /api/ implementacion /setRegistrar/ | Encargado de registrar toda la información y adjuntos en la etapa de implementación por medio del protocolo [HttpPost] |
| /api/ implementacion /setModificar/ | Encargado de modificar toda la información y adjuntos en la etapa de implementación por medio del protocolo [HttpPost] |

***Archivos***

Los archivos y elementos que se adjunten tanto en Registrar como en Modificar serán almacenados en el recurso S3 de Amazon Web Service.

### ApiProyecto : Api/soporte

***Responsabilidad***

Esta interfaz es la encargada de realizar toda la gestión e interacción necesaria cuando los proyectos o iniciativas esten en la etapa de soporte, esto involucra las operaciones principales del http.

***Interfaces – Métodos***

|  |  |
| --- | --- |
| Interfaz | Descripción |
| /api/soporte/ | Consulta todos los proyectos en la etapa de soporte, mediante el protocol [HttpGet] |
| /api/soporte/{id} | Consulta un Proyecto especifico en la etapa de soporte, mediante el Código de identificación del Proyecto y utilizando el protocolo [HttpGet] |
| /api/soporte/setRegistrar/ | Encargado de registrar toda la información y adjuntos en la etapa de soporte por medio del protocolo [HttpPost] |
| /api/soporte/setModificar/ | Encargado de modificar toda la información y adjuntos en la etapa de soporte por medio del protocolo [HttpPost] |

***Archivos***

Los archivos y elementos que se adjunten tanto en Registrar como en Modificar serán almacenados en el recurso S3 de Amazon Web Service.

### ApiProyecto : Api/proyecto

***Responsabilidad***

Esta interfaz es la encargada de realizar toda la gestión, creación y modificación de un proyecto, esto involucra las operaciones principales del http.

***Interfaces – Métodos***

|  |  |
| --- | --- |
| Interfaz | Descripción |
| /api/proyecto/ | Consulta todos los proyectos, mediante el protocol [HttpGet] |
| /api/proyecto/{id} | Consulta un Proyecto especifico, mediante el Código de identificación del Proyecto y utilizando el protocolo [HttpGet] |
| /api/proyecto/setRegistrar/ | Encargado de registrar toda la información por medio del protocolo [HttpPost] |
| /api/proyecto/setModificar/ | Encargado de modificar toda la información por medio del protocolo [HttpPost] |

### White Box *Neuron Software: ApiLoginPermisos*

El siguiente diagrama muestra los principales bloques de construcción del ApiProyecto y como estos se relacionan.



Los seis componentes principales que conforman el ApiLoginPermisos tienen responsabilidades diferentes que se encapsulan dentro del microservicio

|  |  |
| --- | --- |
| Componente | Descripción |
| Servicio | Este componente interno sera el encargado de centralizar todas las funciones y reglas de negocio que se utilicen en el Api, adicionalmente tendra segregación por opción. |
| IServicio | Componente interno encargado de exponer las firma o contratos de los metodos y funciones realizados en la capa servicios a las otras capas |
| Repositorio | Capa encargada de la manipulación y persistencia de datos |
| IRepositorio | Componente interno encargado de exponer las firma o contratos de los metodos y funciones realizados en la capa DAO – Repositorio para ser consumidos por la capa de servicios |
| Domino | Representación de objetos y tablas que se utilizan para la correcta manipulación de los datos |
| Entidades | Representa los objetos de las clases y dominios que se utilizaran en diferentes etapas y capas |

Todos los componentes anteriores son similares para todas las Apis(Microservicios) lo que cambia es el dominio de negocio y el conjunto de operaciones

***Interfaces***

En la siguiente tabla se describen las principales interfaces que expone el ApiLoginPermisos

|  |  |
| --- | --- |
| Interfaz | Descripción |
| /api/usuario/ | Interfaz encargada de la gestión de usuarios, como lo es la creación, modificación |
| /api/rol/ | Interfaz encargada de la creación, modificación y eliminación de roles |
| /api/Login/ | Componente encargado de permitir la autenticación a la aplicación Neuron Software |

A continuación, se detallarán cada una de las interfaces mencionadas anteriormente.

### ApiLoginPermiso : Api/Usuario

***Responsabilidad***

Esta interfaz es la encargada de realizar toda la gestión de los usuarios que podrán ingresar al aplicativo, esto involucra las operaciones principales del http.

***Interfaces – Métodos***

|  |  |
| --- | --- |
| Interfaz | Descripción |
| /api/usuario/ | Consulta todos los usuarios creados, mediante el protocol [HttpGet] |
| /api/usuario/{id} | Consulta un usuario especifico, mediante el Código de identificación del Proyecto y utilizando el protocolo [HttpGet] |
| /api/usuario/setRegistrar/ | Encargado de registrar toda la información básica del usuario, por medio del protocolo [HttpPost] |
| /api/usuario/setModificar/ | Encargado de modificar toda la información del usuario, por medio del protocolo [HttpPost] |
| /api/usuario/setCambiarContra/ | Es la encargada de cambiar la contraseña de un usuario especifico, por medio del protocolo [HttpPost] |

Cada una de las interfaces o metodos mencionados realiza conexión sin tener una dependencia marcada con el ApiEmail, para el envía de notificaciones

***Archivos***

No aplica.

### ApiLoginPermiso : Api/Rol

***Responsabilidad***

Esta interfaz es la encargada de realizar toda la gestión de Roles que se utilizarán en el aplicativo, esto involucra las operaciones principales del http.

***Interfaces – Métodos***

|  |  |
| --- | --- |
| Interfaz | Descripción |
| /api/rol/ | Consulta todos los roles creados, mediante el protocol [HttpGet] |
| /api/rol/{id} | Consulta un rol especifico, mediante el Código de identificación del Proyecto y utilizando el protocolo [HttpGet] |
| /api/rol/setRegistrar/ | Encargado de registrar toda la información necesaria para la creación del rol, por medio del protocolo [HttpPost] |
| /api/rol/setModificar/ | Encargado de modificar la información de un rol especifico, por medio del protocolo [HttpPost] |

***Archivos***

No aplica.

### ApiLoginPermiso : Api/Login

***Responsabilidad***

Esta interfaz es la encargada de realizar la autenticación, ingreso y desconexión en el aplicativo, esto involucra las operaciones principales del http.

***Interfaces – Métodos***

|  |  |
| --- | --- |
| Interfaz | Descripción |
| /api/login/autenticación/ | Encargado de realizar las validaciones correspondientes para ingresar exitosamente a la aplicación. Como lo es usuario y contraseña, rol y permisos. Para esto se utiliza protocolo [HttpPost] |
| /api/login/salir/ | Encargado de destruir las sesiones que se tengan abiertas para el usuario y de cerrar el token de autenticación |

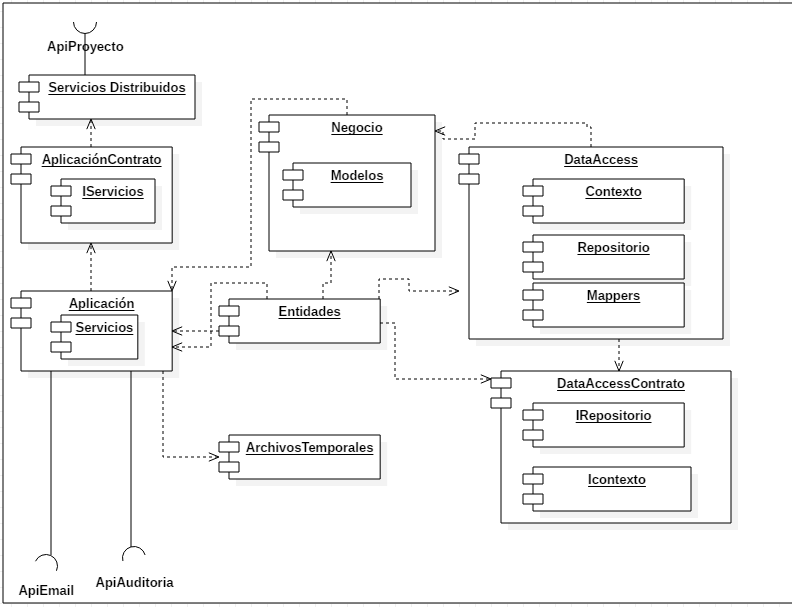
***Archivos***

No aplica.

## Building Blocks Level 2

En los bloques de construcción se detalla la arquitectura interna de las diferentes Apis, aunque en esencia se utilizará la misma arquitectura para todas las Apis

### Building Blocks : ApiProyecto



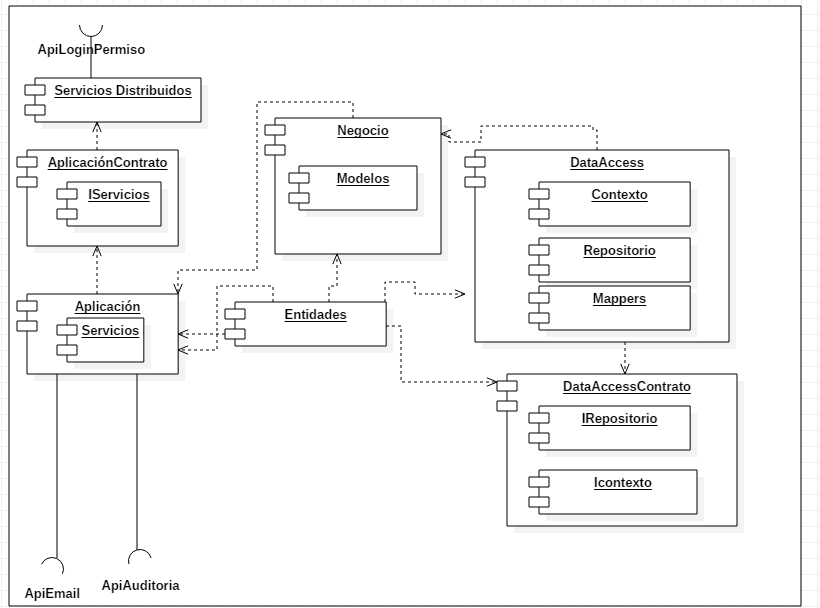
En el diagrama anterior se pueden apreciar los diferentes components y capas que conforman la arquitectura interna.

Mediante la capa de DataAccess se tiene una representación de la base de datos mediante el Contexto y las entidades o tablas que se manejen y en repositorio es toda la logica para la persistencia y manipulación de datos de cada interfaz.

En la capa de aplicación se tendrán todos los servicios que contienen las funcionalidades principales del Api. Adicionalmente desde está capa se llamarán las Apis de email y auditoria, estas apis seran invocadas por medio de eventos, para no tener una dependencia de ninguna clase.

El objeto ArchivosTemporales, es un repositorio de transito o temporal donde se alojaran los adjuntos mientras son cargados en su totalidad al recurso de S3

### Building Blocks : ApiLoginPermisos

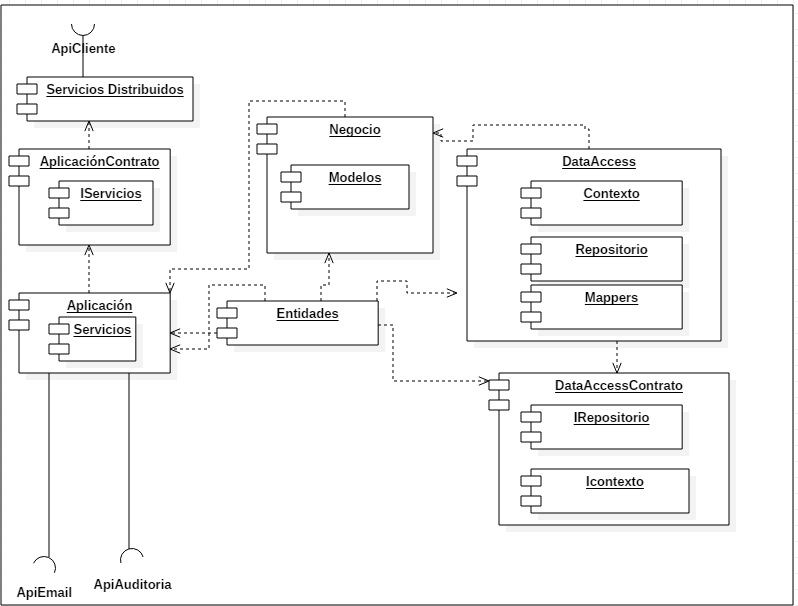


En el diagrama anterior se pueden apreciar los diferentes components y capas que conforman la arquitectura interna.

Mediante la capa de DataAccess se tiene una representación de la base de datos mediante el Contexto y las entidades o tablas que se manejen y en repositorio es toda la logica para la persistencia y manipulación de datos de cada interfaz.

En la capa de aplicación se tendrán todos los servicios que contienen las funcionalidades principales del Api. Adicionalmente desde está capa se llamarán las Apis de email y auditoria, estas apis seran invocadas por medio de eventos, para no tener una dependencia de ninguna clase.

### Building Blocks : ApiCliente



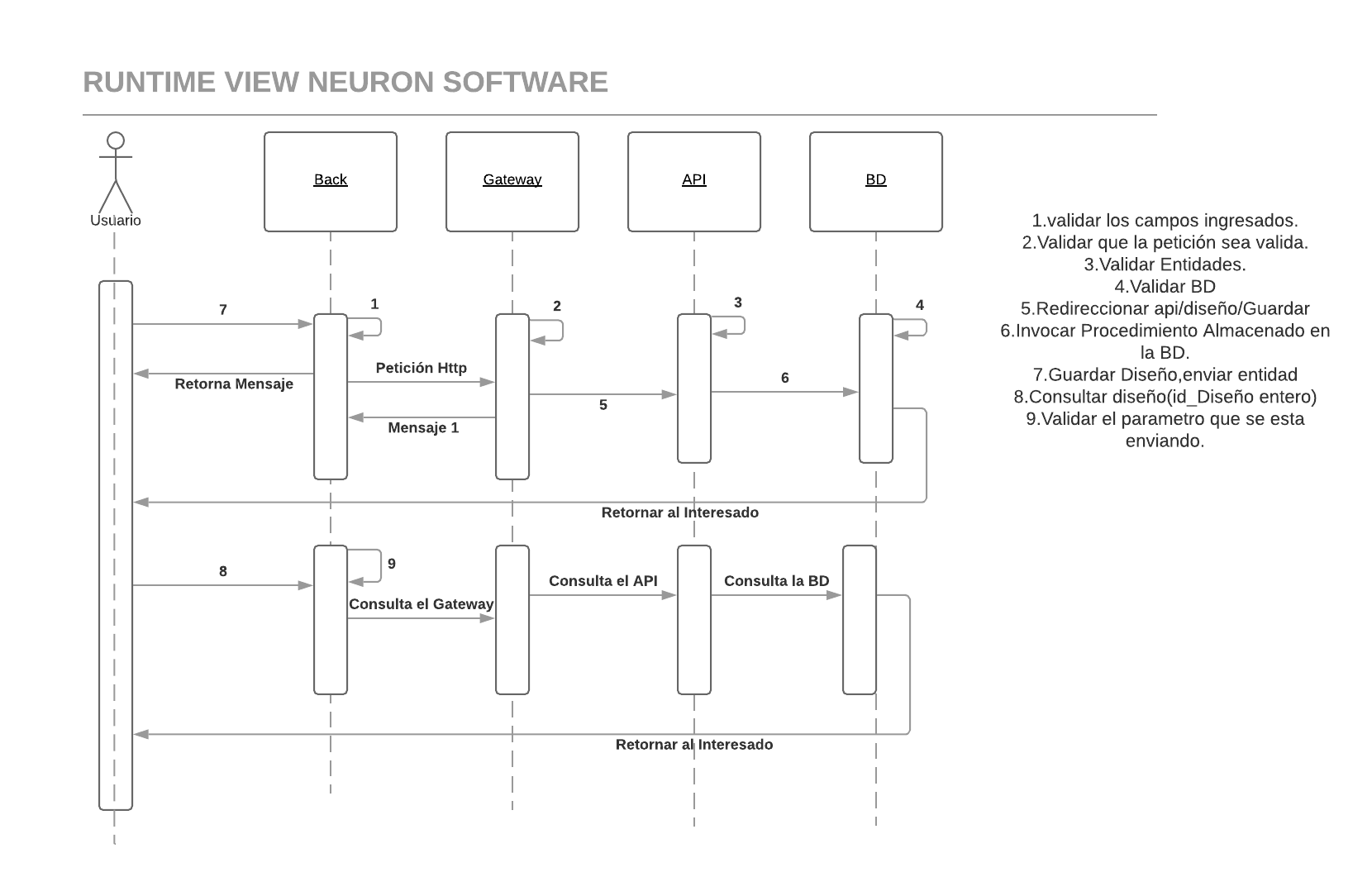
En el diagrama anterior se pueden apreciar los diferentes components y capas que conforman la arquitectura interna.

Mediante la capa de DataAccess se tiene una representación de la base de datos mediante el Contexto y las entidades o tablas que se manejen y en repositorio es toda la logica para la persistencia y manipulación de datos de cada interfaz.

En la capa de aplicación se tendrán todos los servicios que contienen las funcionalidades principales del Api. Adicionalmente desde está capa se llamarán las Apis de email y auditoria, estas apis seran invocadas por medio de eventos, para no tener una dependencia de ninguna clase.

# Runtime View

## Runtime View Diseño

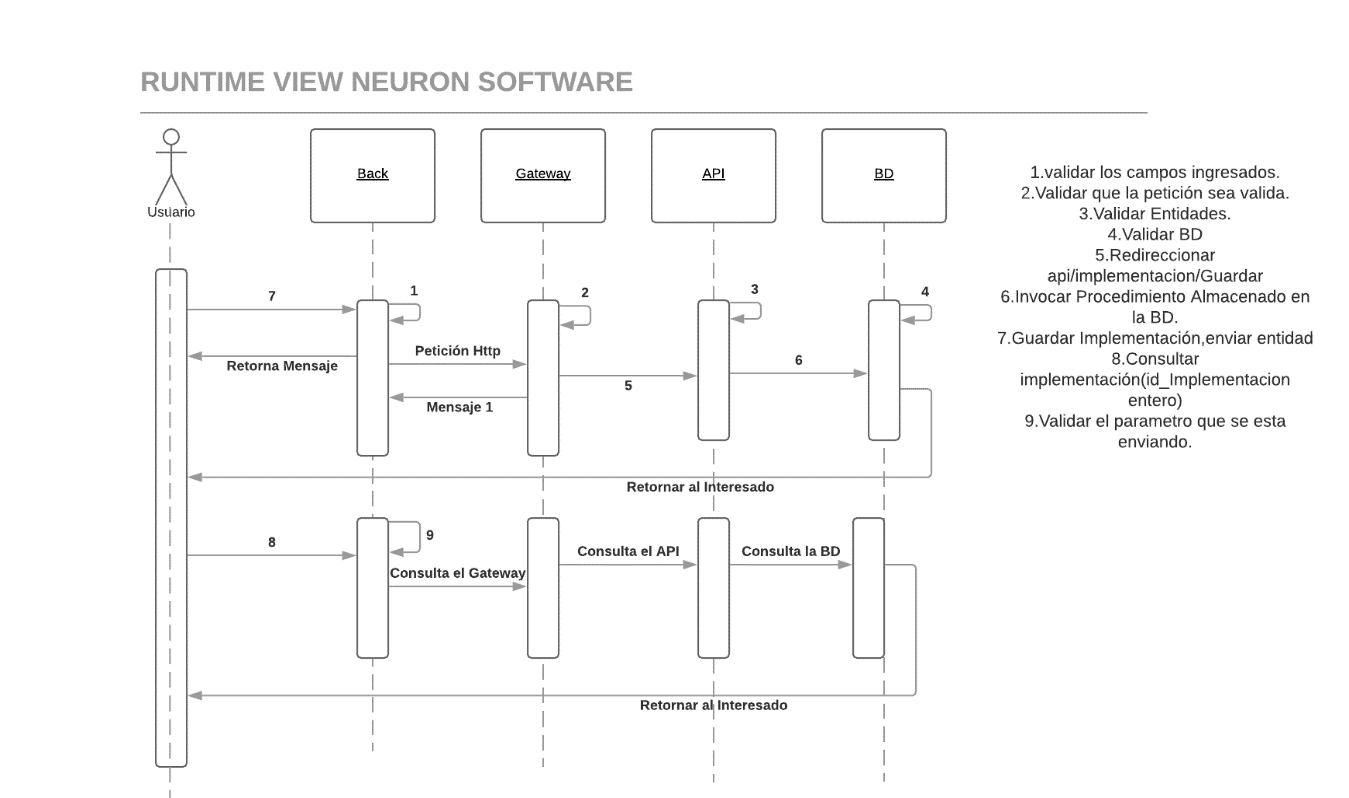


El anterior diagrama donde se muestra el comportamiento y las interacciones básicas del software Neuron

**Cuadro Explicación Runtime View Diseño**

|  |  |
| --- | --- |
| Usuario | Es la persona que ejecuta las diferentes acciones válidas para el sistema en este caso guardar diseño y consultar diseño. El usuario es clasificado por diferentes roles los cuales tienen diferentes tipos de vistas. |
| Backend | Validar que las variables y entidades enviadas por el usuario si estén correctas, si no están correctas devuelve un mensaje indicando que hay un error de tal cosa. |
| ApiGateway | Recibe la solicitud http enviada desde el backend valida que sea una petición válida, previamente existente |
| BD | Recibe la solicitud del api y valida que se esté recibiendo el parámetro adecuado. Este mismo retorna al Usuario |

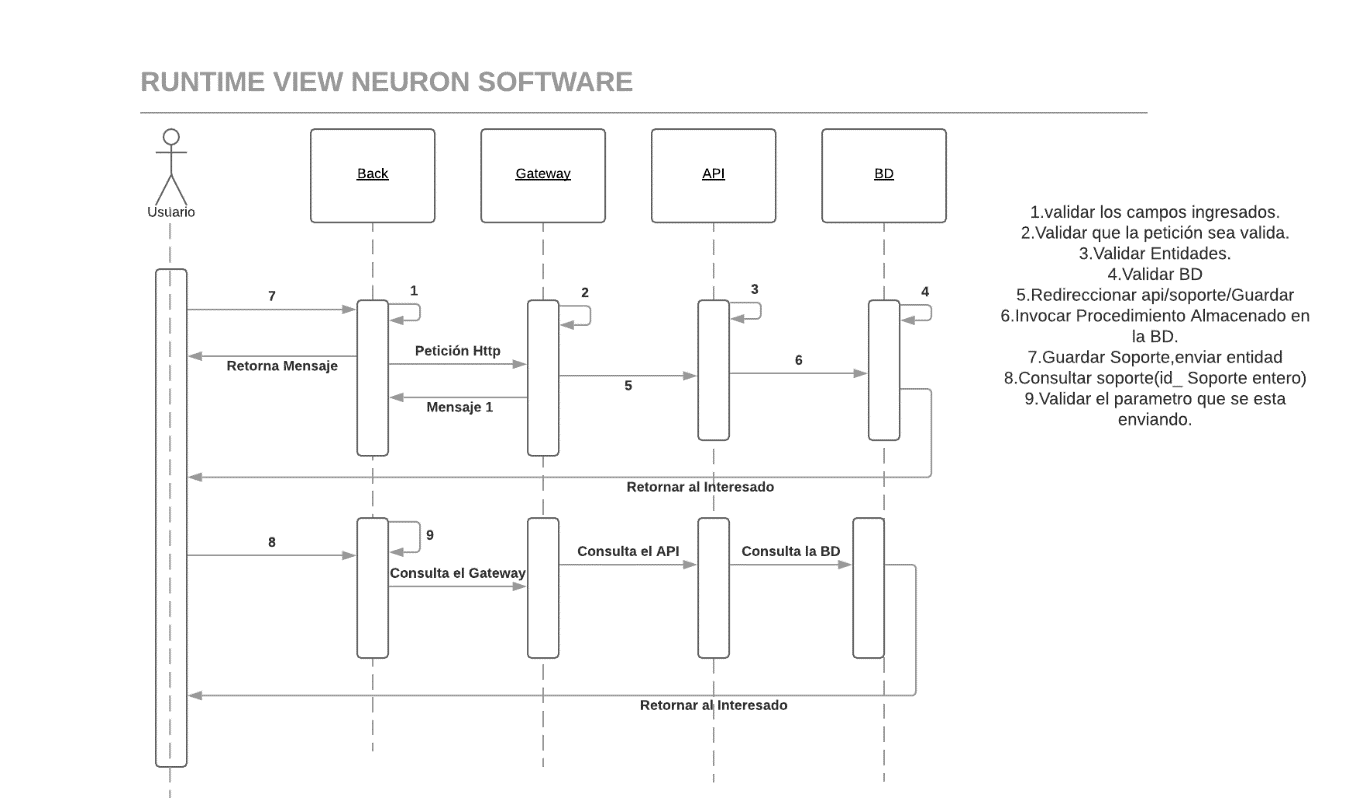
## Runtime View Implementación



|  |  |
| --- | --- |
| Usuario | Es la persona que ejecuta las diferentes acciones válidas para el sistema en este caso guardar implementación y consultar implementación. El usuario es clasificado por diferentes roles los cuales tienen diferentes tipos de vistas. |
| Backend | Validar que las variables y entidades enviadas por el usuario si estén correctas, si no están correctas devuelve un mensaje indicando que hay un error de tal cosa. |
| ApiGateway | Recibe la solicitud http enviada desde el backend valida que sea una petición válida, previamente existente |
| BD | Recibe la solicitud del api y valida que se esté recibiendo el parámetro adecuado. Este mismo retorna al Usuario |

**Cuadro Explicación Runtime View Implementació**

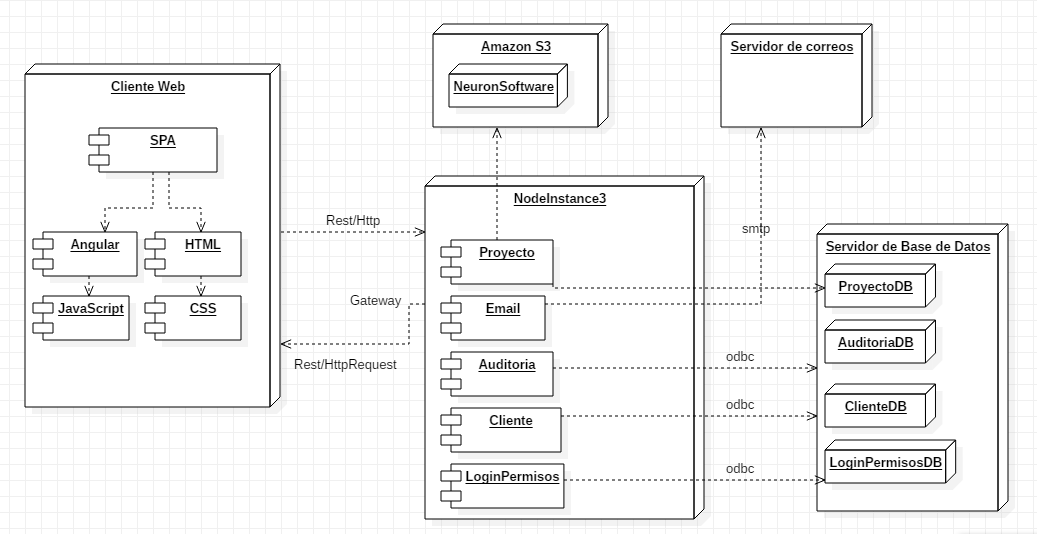
## Runtime View Soporte



|  |  |
| --- | --- |
| Usuario | Es la persona que ejecuta las diferentes acciones validas para el sistema en este caso guardar soporte y consultar soporte. El usuario es clasificado por diferentes roles los cuales tienen diferentes tipos de vistas. |
| Backend | Validar que las variables y entidades enviadas por el usuario si estén correctas, si no están correctas devuelve un mensaje indicando que hay un error de tal cosa. |
| ApiGateway | Recibe la solicitud http enviada desde el backend valida que sea una petición válida, previamente existente |
| BD | Recibe la solicitud del api y valida que se esté recibiendo el parámetro adecuado. Este mismo retorna al Usuario |

**Cuadro Explicación Runtime View Soporte**

# Deployment View



|  |  |
| --- | --- |
| Nodos y Artefactos | Descripción |
| Cliente web | Navegador | Mediante un navegador el usuario final puede visualizer todos los objetos y controles gráficos . En Todos los principales navegadores (Chrome, Firefox, Safari, IE / Edge) deberían funcionar. |
| Kubernetes | nodo 3 | En él se podra manejar y administrar los contenedores donde se encuentran las diferentes Apis |
| Amazon S3 | Es el recurso que se utiliza para el almacenamiento de archivos y otros objetos, que se comunicarán por medio del servicio web |

# Cross-cutting Concepts

## *<Concept 1>*

*<explanation>*

## *Persistency*

Las diferentes bases de datos que se puedan utilizar en el proyecto son de tipo relacional, para realizar el almacenamiento de información en las diferentes bases de datos como también los archivos que se almacenan en el servicio de Amazon S3.

Los accesos a las bases de datos se realizan a través de Entity Framework Core para las Apis Implementadas en .Net, para las Apis implementadas 100% en JavaScript se utiliza Json y ActiveXObject para implementar una conexión.

Cada una de las Apis implementadas contiene un archivo de configuración terminado en [ \_DEV, \_CER, \_ PDN] según el ambiente donde tendrá configurado variables de conexión que se utilicen en dichos ambientes.   
 el acceso y manipulación de datos se realiza por medio de repositorios e interfaces que permitan ser más abiertas y generales.

## *Interfaz de usuario*

La interfaz de usuario predeterminada para Neuron Software que se empaqueta dentro del artefacto final es una Single Page Aplicación única escrita en JavaScript usando Angular JS junto con una plantilla Bootstrap muy predeterminada*.*

## *<Concept n>*

## *Procesamiento de transacciones*

Las transacciones realizadas en la aplicación son asíncronas, de esta forma, una llamada a un proceso de forma asincrónica no tiene que esperar por la respuesta. Cuando le llegue un mensaje, el sistema activa el evento “Recepción de Mensaje”, se da control al código programado para atender al evento y se procesa el mensaje recibido obteniéndose así la respuesta del proceso llamado asincrónicamente.

## *Session Handling*

Las sesiones se controlarán desde las API´s utilizando Json Web Token, el cual permite tener un mayor manejo en las diferentes variables de sesión como lo es tiempo de caducidad, inactividad  
validación del id del token lo que proporciona una mayor seguridad en cada una de las peticiones.

Como la aplicación va tener muchos componentes en JavaScript también se manejarán variables sesiones locales

## *Security*

Neuron Software utiliza API REST que utilizan el protocolo HTTP que admite el cifrado TLS que es un estándar que mantiene privada la conexión de internet y verificar que los datos enviados entre dos sistemas (entre dos servidores, un servidor y un cliente) estén cifrados y no se modifiquen. Eso significa que, si un pirata informático intenta obtener la información de su tarjeta de crédito desde un sitio web de compras, no podrá leer ni modificar sus datos. Para saber si un sitio web está protegido con TLS, solo necesita verificar que la URL comience con "HTTPS" (Protocolo seguro de transferencia de hipertexto).

## *Safety*

Ninguna parte del sistema tiene un aspecto que pone en peligro la vida. Pero, los intrusos maliciosos pueden introducirse en nuestros sistemas mediante la explotación de algunos defectos de software.

## *Communications and Integration*

Neuron Software proporciona un conjunto de API’s que ayudan a realizar integraciones con otros módulos o sistemas, estas API’s exponen todas las funcionalidades requeridas para el sistema, devolviendo un Rest

## [*Plausibility and Validity Checks*](https://biking.michael-simons.eu/docs/index.html#_plausibility_and_validity_checks)

Los tipos de datos y expresiones regulares utilizadas en el sistema Neuron Software son validados mediante DataAnnotation, en algunos casos se utiliza DataAnnotation personalizadas para validar reglas de negocio o estructuras especiales, a nivel de cliente por medio de JavaScript se realizan varias validaciones a nivel de objeto y tipo de datos.

## [*Exception/Error Handling*](https://biking.michael-simons.eu/docs/index.html#_exceptionerror_handling)

Los errores en el manejo de datos inconsistentes (en lo que respecta a la restricción de los modelos de datos), así como las fallas en la validación. Estos errores son manejados por el código del controlador de front-end, permitiendo así que los usuarios puedan visualizar los de mensaje de error o algún tipo de información.   
Estos errores de validaciones funciones o errores técnicos como inconsistencia en el hardware, base de datos y otros son gestionados desde el ApiAuditoria el cual almacena los errores en la base de datos correspondiente

## [*Logging, Tracing*](https://biking.michael-simons.eu/docs/index.html#_logging_tracing)

Los componentes y registros utilizados en la aplicación tienen una configuración inicial la cual se detalla en el número siguiente (8.13).

Adicionalmente no se utiliza ningún complemente o agente externo que facilite el registro y loggin

## *<Concept n>*

## [*Internationalization*](https://biking.michael-simons.eu/docs/index.html#_internationalization)

Para Neuron Software el único idioma admitido es el español, en esta primera versión para una internalización a nivel de idiomas ya que el sistema solo será utilizado al interior de la empresa y de una vicepresidencia especifica.

## *Migration*

El proyecto Neuron Software no nace como una migración o actualización de un sistema, por lo contrario nace de la necesidad de mejorar procesos y ser más eficientes en estos utilizando tecnología de punta (JavaScript, Angular )

## *Testability*

Se hicieron Pruebas de Stress y de Rendimiento

# Design Decisions

**Utilizar bases de datos históricas para mejorar el performance en reporteria**

**Problema**

Neuron Software es un sistema de trazabilidad de iniciativas y proyectos a través de diferentes etapas, lo cual requiere consultar y almacenar información del cambio de estado de los mismos, la información de los proyectos debe ser persistente en el tiempo, aun así, el proyecto ya se encuentre cerrado, por regulación la información debe poderse consultar mínimamente 10 años atrás, pero esto a la vez afecta directamente las tablas transaccionales del día a día.

**Restricciones**

* El reporte de proyectos o iniciativas no debe durar más de 30 segundos.
* Se debe sacrificar rendimiento por seguridad por que la disponibilidad se debe priorizar en el tiempo

**Supuestos**

* La información que se descargue de los proyectos históricos debe tener todo un set de datos congruentes.

**Decisión**

Para evitar que las tablas transacciones del día a día se empiecen a llenar exponencialmente afectando considerablemente el performance de las operaciones, se diseñará una base de datos encargada de almacenar la información histórica, que consiste en tener un proceso lambda, el cual se ejecutará cada 6 meses y tomará la información de las tablas transaccionales de la basa de datos de Neuron Software y la pasara a una nueva base de datos de histórico dejando en la base de datos del día a día información de los dos últimos años, por otro lado tanto en la base de datos de histórico como en la del día a día se implementarán varios índices que ayuden a la optimización del proceso. Al realizar esto se libera espacio, mejorando el performance en las operaciones del día a día, así mismo como la respuesta en los diferentes informes.

**Información temporal o en cookies**

**Problema**

En repetidas ocasiones durante todo el proceso el sistema utiliza cierta información que es repetitiva, donde constantemente tiene que ir a obtener dicha información en la base de datos ocasionando una latencia por cada conexión que se abra.

**Supuestos**

* la información básica del usuario y del proyecto se vuelve repetitiva para la gran mayoría de procesos.

**Decisión**

para poder trabajar con la información que en la gran mayoría de procesos se repite se implementará un sistema de datos temporales cookies, esto por medio de la base de datos ElastiCache de Amazon que trabaja con información en memoria, implementando esto la latencia se disminuye considerablemente a microsegundos y no sería necesario abrir tantas conexiones directas a la base de dato

Al implementar las anteriores decisiones el Sistema ganaria en performace pero se sacrificaria una parte en seguridad.

# Quality Requirements

## Quality Scenarios

|  |  |
| --- | --- |
| **Atributo** | Desempeño (Eficiencia) |
| **Fuente del evento** | Usuarios del sistema, procesos internos |
| **Estimulo** | Intentar consultar o guardar información con un tiempo de procesamiento elevado |
| **Ambiente** | Sistema Neuron Software |
| **Artefacto** | Interacción con la base de datos como consultar, guardar, modificar y otros recursos que conlleven a un procesamiento. |
| **Respuesta** | El sistema deberá realizar el procesamiento de datos tanto de entrada como de salida y cargar interfaces en un tiempo optimo y sin tener ninguna afectación por latencia |
| **Medición** | Todas las peticiones que realice el sistema a la base de datos no deben tener una latencia superior a 30 segundos, las interfaces deben cargar mínimo en 5 segundos. |

|  |  |
| --- | --- |
| **Atributo** | Interoperabilidad |
| **Fuente del evento** | Usuarios del sistema, recursos y procesos internos |
| **Estimulo** | Proceso complejo y poco estandarizado para recibir y exponer información con otras fuentes, generando reprocesos |
| **Ambiente** | Sistema Neuron Software |
| **Artefacto** | Interacción con otros sistemas o fuentes que requieren datos de Neuron Software o sistemas que expongan algún recurso utilizado por Neuron |
| **Respuesta** | El sistema deberá contar con interfaces que permitan integrar fácilmente con otros sistemas manejando cierto grado de estandarización |
| **Medición** | Todas las integraciones que se realicen deben tener una estructura similar o algo estándar que permita poder incluir o excluir fácilmente la integración con otros componentes. |

|  |  |
| --- | --- |
| **Atributo** | Portabilidad |
| **Fuente del evento** | Usuarios del sistema, recursos y procesos internos |
| **Estimulo** | Poca capacidad para ejecutar tareas o levantar todo el sistema en arquitecturas y sistemas operativos diferentes a los iniciales, generando dependencia tecnológica. |
| **Ambiente** | Sistema Neuron Software |
| **Artefacto** | Interacción con otros sistemas o fuentes que requieren datos de Neuron Software o sistemas que expongan algún recurso utilizado por Neuron |
| **Respuesta** | El sistema deberá contar con la capacidad de funcionar y levantar todo el sitio en otras plataformas, sin tener un sobresfuerzo mayor abarcando todo el concepto multiplataforma. |
| **Medición** | El sistema Neuron Software deberá ser de fácil despliegue en cualquier tipo de plataforma operativa con la misma funcionalidad en todas las plataformas. |

|  |  |
| --- | --- |
| **Atributo** | Escalabilidad |
| **Fuente del evento** | Usuarios del sistema, procesos y recursos de infraestructura |
| **Estimulo** | Realizar un crecimiento del sistema tanto vertical como horizontal sin tener grandes impactos. |
| **Ambiente** | Sistema Neuron Software |
| **Artefacto** | Creación de nuevos módulos o funcionalidades que permitan un crecimiento exponencial de la aplicación, de igual manera que permita escalar la infraestructura en picos de uso muy altos. |
| **Respuesta** | El sistema deberá ser modular y apificado para tener un crecimiento adecuado sin afectar otras funcionalidades, contando con contenerización para escalar la infraestructura cuando sea necesario. |
| **Medición** | Crecimiento funcional y a nivel de infraestructura cuando la demanda y las condiciones lo requieran sin tener una afectación grande. |

|  |  |
| --- | --- |
| **Atributo** | Fiabilidad |
| **Fuente del evento** | Usuarios del sistema, procesos y recursos de infraestructura |
| **Estimulo** |  |
| **Ambiente** | Sistema Neuron Software |
| **Artefacto** | Creación de nuevos módulos o funcionalidades que permitan un crecimiento exponencial de la aplicación, de igual manera que permita escalar la infraestructura en picos de uso muy altos. |
| **Respuesta** | El sistema deberá ser modular y apificado para tener un crecimiento adecuado sin afectar otras funcionalidades, contando con contenerización para escalar la infraestructura cuando sea necesario. |
| **Medición** | Crecimiento funcional y a nivel de infraestructura cuando la demanda y las condiciones lo requieran sin tener una afectación grande. |

|  |  |
| --- | --- |
| **Atributo** | Seguridad |
| **Fuente del evento** | Usuarios del sistema |
| **Estimulo** | Intentar acceder a recursos u opciones de otro rol |
| **Ambiente** | Sistema Neuron Software |
| **Artefacto** | Parametrización general e información de proyectos |
| **Respuesta** | El sistema debe avisar y controlar un ingreso invalido o no autorizado |
| **Medición** | El sistema debe mostrar un acceso restringido ante alguna operación invalida |
|  |  |
|  |  |
| **Atributo** | Usabilidad |
| **Fuente del evento** | Usuarios del sistema |
| **Estimulo** | Pasos numerosos o inadecuados para realizar procesos en el sistema y look and feel fuera de los lineamientos empresariales |
| **Ambiente** | Sistema Neuron Software |
| **Artefacto** | Interacción con cada uno de los objetos gráficos del sistema Neuron Software |
| **Respuesta** | El sistema debe permitir realizar cada uno de los procesos de forma simple e intuitiva minimizando la cantidad de pasos |
| **Medición** | El sistema debe ser amigable, intuitivo y fácil de utilizar |

# Risks and Technical Debts

Los riesgos técnicos en el proyecto Neuron Software lo dividimos en dos etapas principales (implementación y proceso productivo)

**Implementación**

* Un riesgo latente en esta etapa es el conocimiento en las herramientas y tecnologías a utilizar, que de una o de otra forma impacta la implementación del proyecto, para contra restar esto se harán alineaciones técnicas de conocimiento.
* Tanto la implementación como el diseño de la solución juegan un papel muy importante, y es uno de los riesgos más críticos en esta etapa, como garantizar que efectivamente se está implementado lo que se definió en el diseño de la solución y que el diseño se pueda adaptar conforme a otras necesidades, para esto se deberá estandarizar la forma de realizar la implementación ayudados por su parte con revisiones par.

**Productivo**

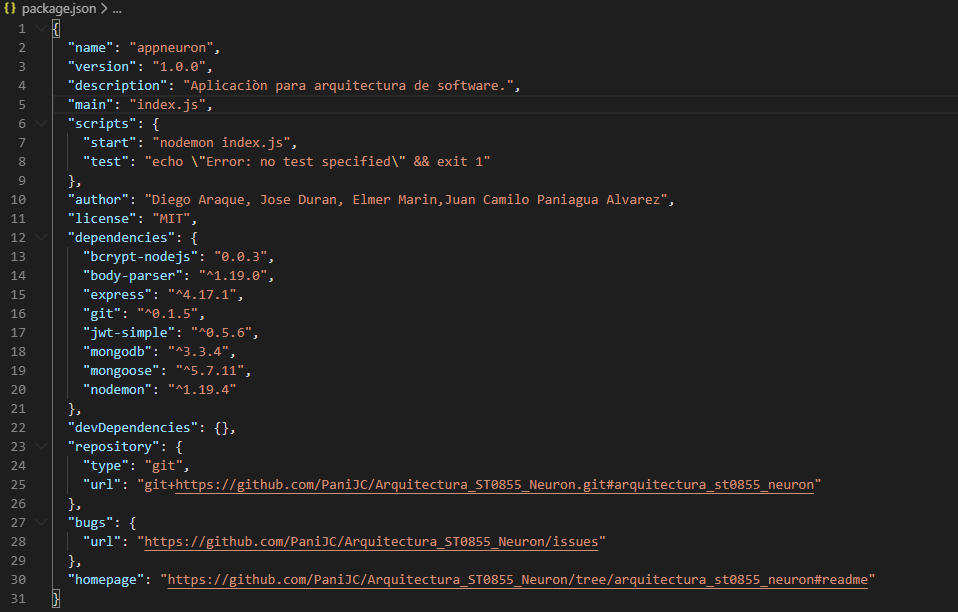
* Uno de los principales riesgos que se pueden presentar en producción, es el rendimiento de la aplicación en picos elevados, para reducir esto se implementará un balanceador de carga con un auto scaling donde se podrá tener un crecimiento a nivel de infraestructura en picos elevados al tiempo que se tendrá que optimizar consultar y procesos en base de datos para que el performance no se vea afectado.
* Puede existir un riesgo en cuento los datos, es la posibilidad que la base de datos o instancia sufra algún apagón o daño, para esto se tendrán programados backups a las diferentes bases de datos de forma periódica, teniendo en cuenta un tiempo tolerable en el que el negocio podría aguantar sin la información.

# Glossary

|  |  |
| --- | --- |
| Termino | Descripción |
| AngularJS | es un marco de aplicaciones web de código abierto mantenido principalmente por Google y por una comunidad de desarrolladores y corporaciones individuales para abordar muchos de los desafíos encontrados en el desarrollo de aplicaciones de una sola página. |
| Json | Es un formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de  objetos de JavaScript, aunque, debido a su amplia adopción como alternativa a xml se considera (año [2019](https://es.wikipedia.org/wiki/2019)) un formato independiente del lenguaje |
| APIs | Interfaz de programación de aplicaciones, es un conjunto de [subrutinas](https://es.wikipedia.org/wiki/Subrutina), funciones y procedimientos (o [métodos](https://es.wikipedia.org/wiki/M%C3%A9todo_(inform%C3%A1tica)), en la [programación orientada a objetos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos)) que ofrece cierta [biblioteca](https://es.wikipedia.org/wiki/Biblioteca_(programaci%C3%B3n)) para ser utilizado por otro [software](https://es.wikipedia.org/wiki/Software) como una [capa de abstracción](https://es.wikipedia.org/wiki/Capa_de_abstracci%C3%B3n) |
| API GateWay | es la pieza encargada de unificar y administrar la publicación de APIs para que sean consumidas por otras aplicaciones o por los desarrolladores |
| JavaScript | es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como [orientado a objetos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos),[3](https://es.wikipedia.org/wiki/JavaScript#cite_note-3)​ [basado en prototipos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_basada_en_prototipos), [imperativo](https://es.wikipedia.org/w/index.php?title=Programaci%C3%B3n_imperativa&action=edit&redlink=1), débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente |
| Bootstrap | Bootstrap es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios. |
| Amazon S3 | Es un servicio de almacenamiento de objetos que ofrece escalabilidad, disponibilidad de datos y seguridad en la nube |
| SPA | Single Page Application, es un sitio web que cabe en una sola página con el propósito de dar una experiencia más fluida a los usuarios como una aplicación de escritorio |

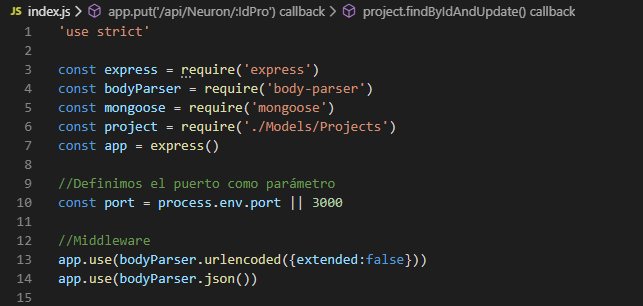
# Snippets y Evidencias.

## Package del proyecto



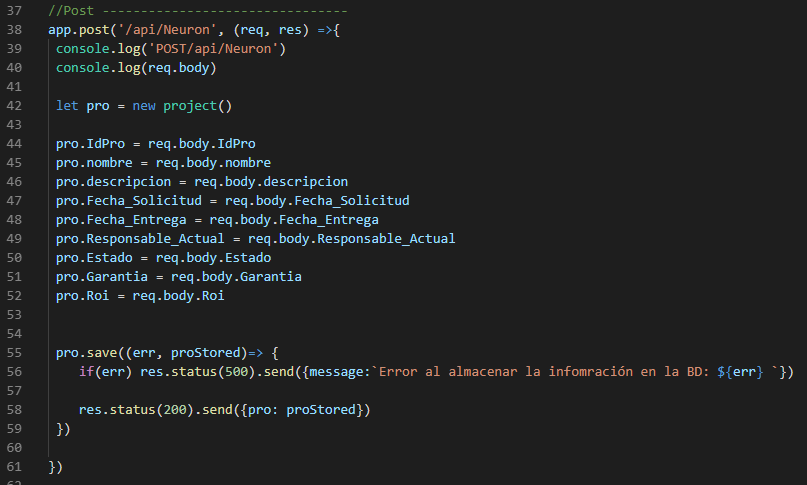
## API Principal (api/Neuron)

Definición de puerto, configuración de Server, Middleware e importación de librerías:

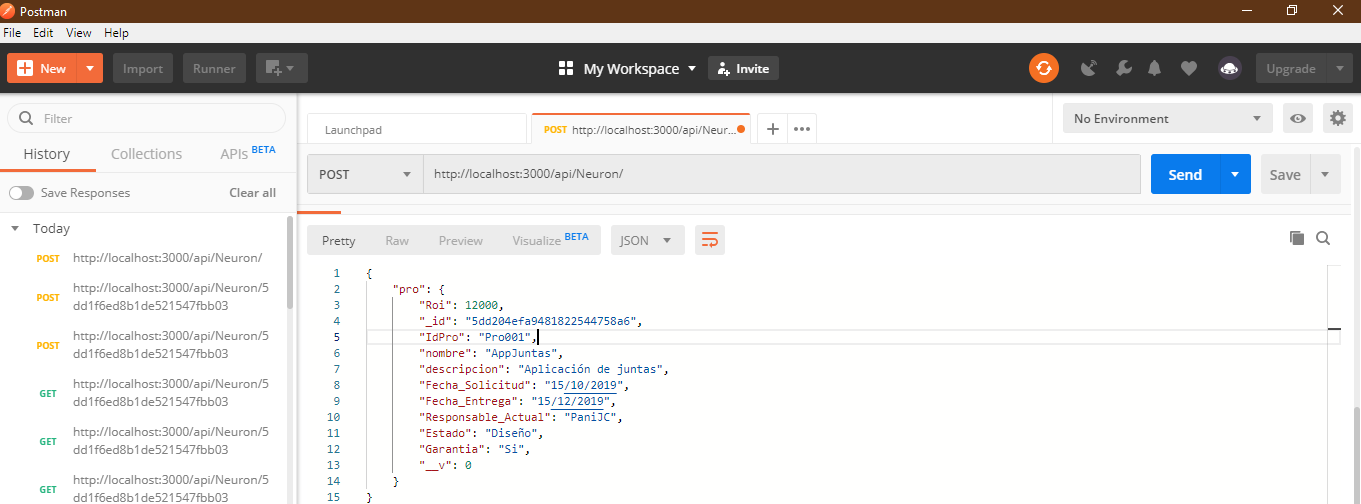


### Definición POST

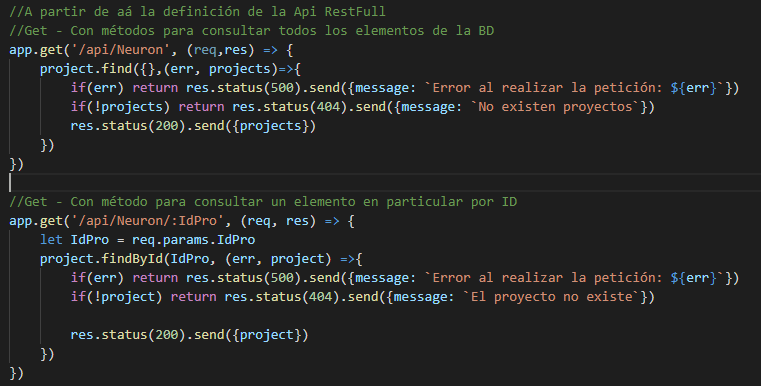
**Definición POST**



*Pruebas realizadas en POSTMAN*

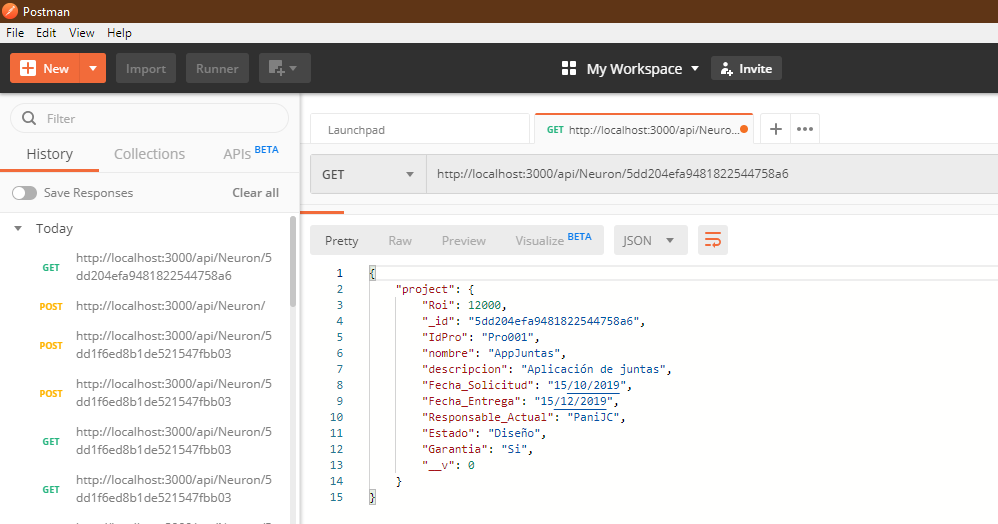


### Definición GET

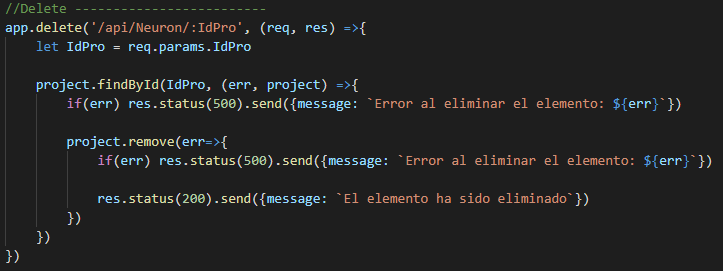


*Pruebas realizadas en POSTMAN*

Este get aplicado en el ejemplo corresponde al get de un elemento en particular. En el ejemplo del DELETE se puede ver el get de todos los elementos existentes.



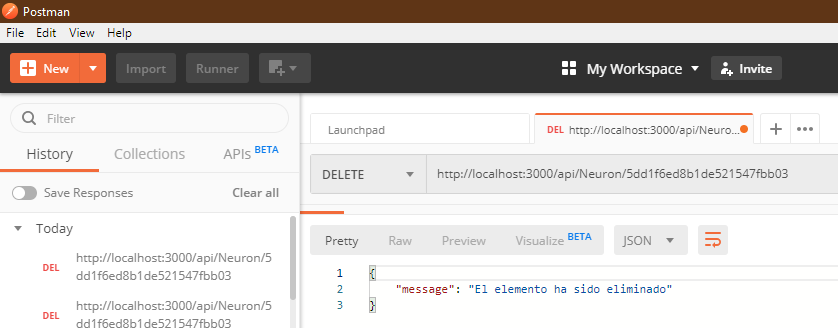
### Definición DELETE



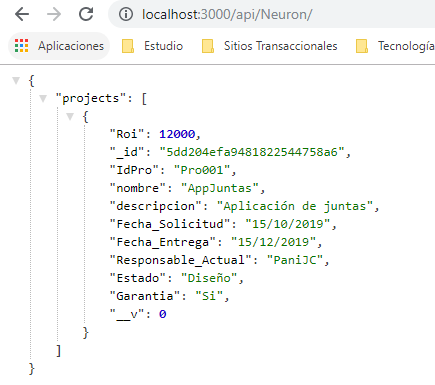
Creamos previamente dos registros (tienen diferente id) y los consultamos con el GET para todos los elementos:



*Pruebas realizadas en POSTMAN*

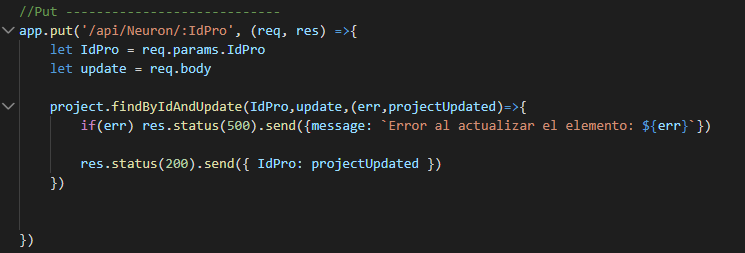


En la gráfica que sigue podemos ver que solo queda un elemento:



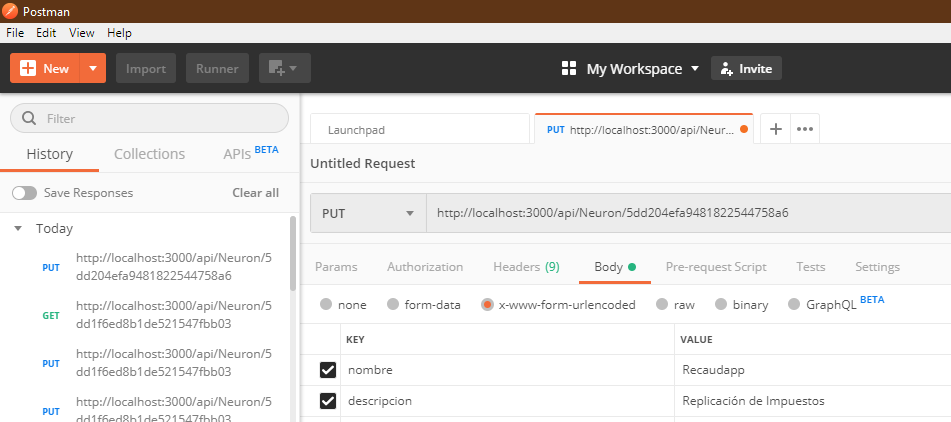
### Definición PUT

**Definición PUT**



*Pruebas realizadas en POSTMAN*

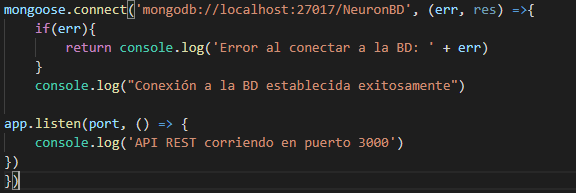
Solo actualizamos los campos nombre y descripción



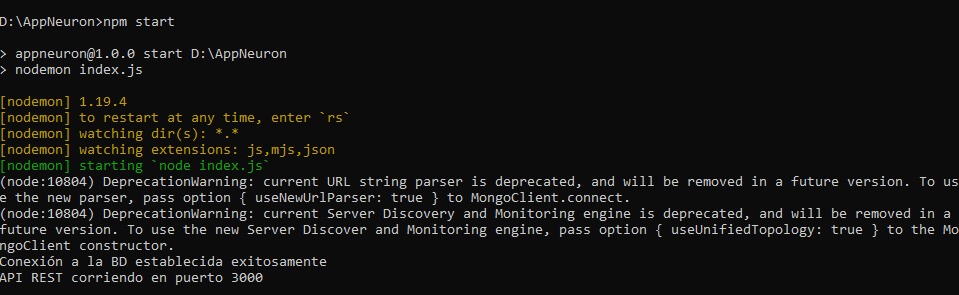


## Configuración de Conexión a MONGO DB

Configuración de conexión a BD(Mongo) y manejo de errores:

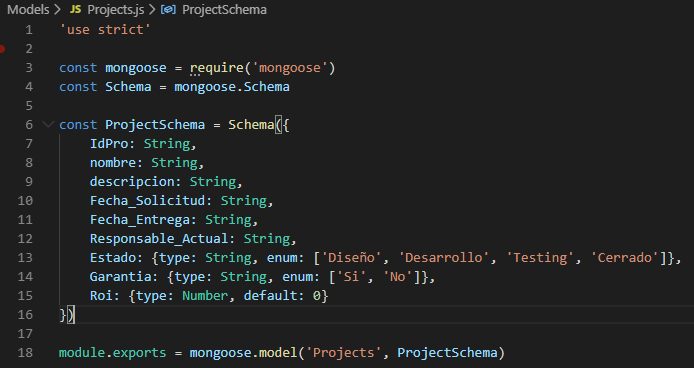


*Pruebas realizadas en NODE:*



## Módulos

### Módulo BD – ‘Projects’



### Módulo BD – “User” (Autenticación por Token)



## Nota FINAL – IMPORTANTE

Es importante aclárale al profesor que para lograr presentar un trabajo orientado al tema de los microservicios y a un modelo de arquitectura esperado, cambiamos la arquitectura del proyecto en la que estábamos trabajando, ya que las bases de datos eran relacionales (SQL) mientras que las nuevas son NO Relacionales (MONGO DB) y la arquitectura aplicada en el nuevo modelo cumple con las métricas de SOLID, Single Page Application, MVC, Autenticación por Token. Fue un aprendizaje de muy alta importancia para nosotros. Prácticamente nos tocó “desaprender” la manera en la que programábamos con modelos monolíticos y estructuras deficientes, a esto que nos enseñó, dinámico, escalable, valioso de verdad.