

**UNIVERSIDAD TECNOLÓGICA ISRAEL**

**MEMORIA DEL PROYECTO DE FIN DE CARRERA**

**INGENIERÍA EN SISTEMAS INFORMÁTICOS**

**TEMA:**

MODULO BACKEND (API) PARA SISTEMA ELECCIÓN DE CANDIDATOS

**AUTOR:**

DANNY ALEXANDER CÁRDENAS HIDALGO

**TUTOR:**

MG. LUIS FERNANDO AGUAS B.

**QUITO, ECUADOR**

**2021**

**UNIVERSIDAD TECNOLÓGICA ISRAEL**

**APROBACIÓN DEL TUTOR**

En mi calidad de Tutor del Trabajo de Titulación certifico:

Que el trabajo de titulación **“MODULO BACKEND (API) PARA SISTEMA ELECCIÓN DE CANDIDATOS.”**, presentado por DANNY ALEXANDER CÁRDENAS HIDALGO estudiante de la Carrera Ingeniería en Sistemas Informáticos, reúne los requisitos y méritos suficientes para ser sometido a la evaluación del Tribunal de Grado, que se designe, para su correspondiente estudio y calificación.

Quito 09/05/2021

TUTOR

----------------------------------------

Mg. Luis Fernando Aguas B.

**TABLA DE CONTENIDOS**

[INTRODUCCIÓN i](#_Toc8329311)

[Antecedentes de la situación objeto de estudio i](#_Toc8329312)

[Planteamiento del problema i](#_Toc8329313)

[Justificación ii](#_Toc8329314)

[Objetivos iii](#_Toc8329315)

[General iii](#_Toc8329316)

[Objetivos específicos iii](#_Toc8329317)

[Alcance iii](#_Toc8329318)

[1 CAPÍTULO 1. PROPUESTA 5](#_Toc8329319)

[1.1 Diagramas de procesos 5](#_Toc8329320)

[1.2 Especificación de requerimientos 9](#_Toc8329321)

[1.2.1 Ámbito del software 9](#_Toc8329322)

[1.2.2 Funciones del producto 10](#_Toc8329323)

[1.2.3 Características de los usuarios del sistema 21](#_Toc8329324)

[1.2.4 Restricciones 22](#_Toc8329325)

[1.2.5 Requisitos 23](#_Toc8329326)

[2 CAPÍTULO 2. RESULTADOS 25](#_Toc8329327)

[2.1 Diseño general 25](#_Toc8329328)

[2.2 Esquema de la base de datos (SGBDD) 25](#_Toc8329329)

[2.3 Diagrama de la arquitectura del sistema 26](#_Toc8329330)

[2.4 Diseño de interfaces 28](#_Toc8329331)

[2.5 Estándares de programación utilizados 29](#_Toc8329332)

[2.6 Pruebas 31](#_Toc8329333)

[2.7 Implementación 32](#_Toc8329334)

[2.7.1 Requerimientos de hardware y software 32](#_Toc8329335)

[3 CONCLUSIONES 34](#_Toc8329336)

[4 RECOMENDACIONES 35](#_Toc8329337)

[5 REFERENCIAS BIBLIOGRÁFICAS 36](#_Toc8329338)

[6 ANEXOS 1](#_Toc8329339)

**LISTA DE FIGURAS**

[Figura 1. Proceso actual, no automatizado de un evento genérico al elegir candidatos. 6](#_Toc73279524)

[Figura 2. Diagrama del proceso automatizado con el software ElectionSystem. 8](#_Toc73279525)

[Figura 3. Tabla de base de datos principal 26](#_Toc73279526)

[Figura 4. Arquitectura del Sistema 28](#_Toc73279527)

[Figura 5. Vista General de Pantalla Swagger 29](#_Toc73279528)

[Figura 6. Vista de muestra del método login (swagger) 29](#_Toc73279529)

[Figura 7. Vista de muestra del método login en caso de ser correcta la petición (swagger) 29](#_Toc73279530)

[Figura 8. Vista de muestra del método login en caso de ser errónea la petición (swagger) 29](#_Toc73279531)

**LISTA DE TABLAS**

[Tabla 1. Historias de Usuario: Creación de usuarios 11](#_Toc73279585)

[Tabla 2. Historias de Usuario: Permitir el acceso al sistema 11](#_Toc73279586)

[Tabla 3. Historias de Usuario: Permitir al administrador Crear eventos 12](#_Toc73279587)

[Tabla 4. Historias de Usuario: Restringir la cantidad de eventos 12](#_Toc73279588)

[Tabla 5. Historias de Usuario: Permitir agregar más administradores al evento 13](#_Toc73279589)

[Tabla 6. Historias de Usuario: Permitir Registrar candidatos al evento 13](#_Toc73279590)

[Tabla 7. Historias de Usuario: Permitir Registrar participante al evento 14](#_Toc73279591)

[Tabla 8. Historias de Usuario: Permitir Registro masivo participante al evento 14](#_Toc73279592)

[Tabla 9. Historias de Usuario: Permitir Registro por parte del participante 15](#_Toc73279593)

[Tabla 10. Historias de Usuario: Permitir dar de baja al evento 15](#_Toc73279594)

[Tabla 11. Historias de Usuario: Permitir dar de baja candidato por evento 16](#_Toc73279595)

[Tabla 12. Historias de Usuario: Permitir dar de baja participante por evento 16](#_Toc73279596)

[Tabla 13. Historias de Usuario: Permitir actualizar Datos del evento 17](#_Toc73279597)

[Tabla 14. Historias de Usuario: Permitir Actualizar datos del usuario 17](#_Toc73279598)

[Tabla 15. Historias de Usuario: Permitir Actualizar datos del candidato 18](#_Toc73279599)

[Tabla 16. Historias de Usuario: Permitir a los participantes votar 18](#_Toc73279600)

[Tabla 17. Historias de Usuario: Los registros y votaciones deberán tener fechas de control. 19](#_Toc73279601)

[Tabla 18. Historias de Usuario: Dar seguridad al login mediante tokens 19](#_Toc73279602)

[Tabla 19. Historias de Usuario: Permitir subir imágenes al servidor 20](#_Toc73279603)

[Tabla 20. Historias de Usuario: Notificar por correo diferentes acciones. 20](#_Toc73279604)

[Tabla 21. Historias de Usuario: Implementar servicio para olvidó contraseña 21](#_Toc73279605)

[Tabla 22. Perfiles de usuario 22](#_Toc73279606)

[Tabla 23. Requerimientos Funcionales 23](#_Toc73279607)

[Tabla 24. Requerimientos no Funcionales 24](#_Toc73279608)

[Tabla 25. Tabla de Scrum 25](#_Toc73279609)

[Tabla 26. Arquitectura del Sistema 26](#_Toc73279610)

[Tabla 27. Estándares de programación utilizados 30](#_Toc73279611)

# INTRODUCCIÓN

## ****Antecedentes de la situación objeto de estudio****

Las elecciones de candidatos ya sea para escoger reinas, presidentes, funcionarios u otros cargos en un grupo determinado, son actividades frecuentes que se las realizaba con la participación presencial de cada uno de los individuos que tenían la obligación de escoger un representante el grupo correspondiente.

En el ecuador a diario se realizan a diario se realizan elecciones pequeñas ya sea en universidades, parroquias, pueblos, o diferentes grupos establecidos que buscan seleccionar a una persona para determinado puesto, actualmente con la pandemia mundial que atraviesa la sociedad es prohibido realizar estas reuniones que permitirían a los individuos votar o seleccionar un candidato en ceremonias presenciales. (Robles, 2010)

## ****Planteamiento del problema****

La problemática principal surge con las restricciones que exige la pandemia mundial que atravesamos, las cuales obliga a evitar en lo posible un contacto físico con muchas personas y por lo tanto cancelar los eventos que atraigan varios espectadores o público en general a un determinado lugar.

Con el problema antes mencionado, principalmente los eventos de elecciones de candidatos o reinas son cancelados y muy difícilmente manejables mediante chats comunes, u otro medio que no implemente las características específicas que requieren estos concursos, como puede ser: presentar información de candidatos ya sea características, imágenes, videos, propuestas etc, llevar un control de los individuos que realizar sus votos y también presentar las estadísticas al los participantes para conocer los resultados.

## ****Justificación****

El propósito de desarrollar un API para elecciones de candidatos tiene como objetivo atacar 2 fines directamente:

1) Poder crear un administrador para eventos de selección de candidatos que ayude a los grupos, instituciones o barrios que requieran realizar un evento para escoger representantes de algún tipo específico, evitando de esta manera las aglomeraciones de personas en lugares concentrados y también facilitando la información de los candidatos y sus propuestas o características, a público final que hará la selección.

2) Al crear un API se pretende abrir la posibilidad a ser implementado un cualquier visualizador (FrontEnd) ya sea dispositivos móviles, aplicaciones web, aplicaciones de escritorio, etc. Dichos visualizadores no tendrán que preocuparse por la lógica de cómo se lleva el concurso, simplemente deben preocuparse por presentar un diseño agradable a los usuarios finales y el resto de trabajo será solicitudes y respuestas HTTP.

## ****Objetivos****

### ****General****

### Desarrollo de un Backend (API) que permita gestionar el proceso de elecciones de candidatos.

### ****Objetivos específicos****

## Aplicar la metodología Scrum para llevar un control correcto y rápido en el desarrollo del módulo.

## Diseñar los modelos y esquemas que permitan fluir la lógica que conlleva los eventos de selecciones.

## Implementar seguridades a todas las peticiones realizadas al API mediante cualquier implementación de FrontEnd.

## Exponer todos los servicios que estén disponibles en el API mediante Swagger para su correcto uso y consumo por parte de terceros.

## ****Alcance****

El proyecto tendrá como alcance la creación de un API consumible, implementado en un entorno de desarrollo, para elecciones de candidatos con proyección y soporte a una cantidad variable de participantes dentro de los eventos a realizarse en empresas o instituciones educativas. El software permitirá nombrar un candidato ganador, dentro de un grupo de participantes en el evento; a continuación, se detalla los módulos a entregar:

• Autentificación y autorización.

• Registros de Candidatos.

• Registro de participantes.

• Votación de los participantes registrados.

En el proceso de Autentificación y autorización se receptará los datos proporcionados por el usuario y se los validará en el servidor para verificar que sean correctos, entonces podrá acceder a los demás recursos o servicios.

En el proceso de Registro de candidatos los usuarios participantes podrán registrarse en las pantallas correspondientes y estos datos serán enviados al servidor para validarlos y concluir con el registro.

En el proceso de registro de participantes: se podrá subir un archivo con la lista de los usuarios que podrán asistir al evento determinado, esta lista la analizará el servidor y la guardará.

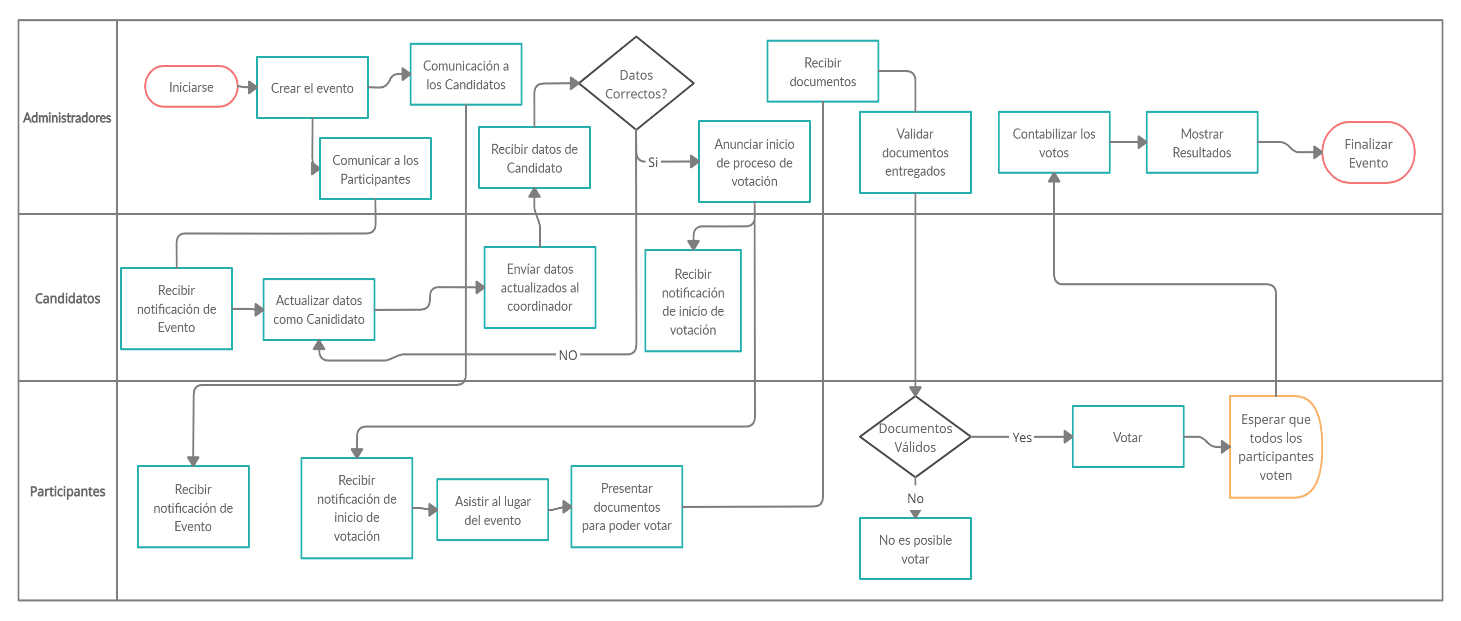
Para el módulo de votación: los usuarios ingresarán a las pantallas correspondientes y podrán elegir solo a un candidato para votar, este voto será registro por el servidor y almacenado en una base de datos.

El módulo solamente contará con una infraestructura para soportar eventos con una cantidad de personas correspondiente a instituciones educativas o barrios.

# CAPÍTULO 1. PROPUESTA

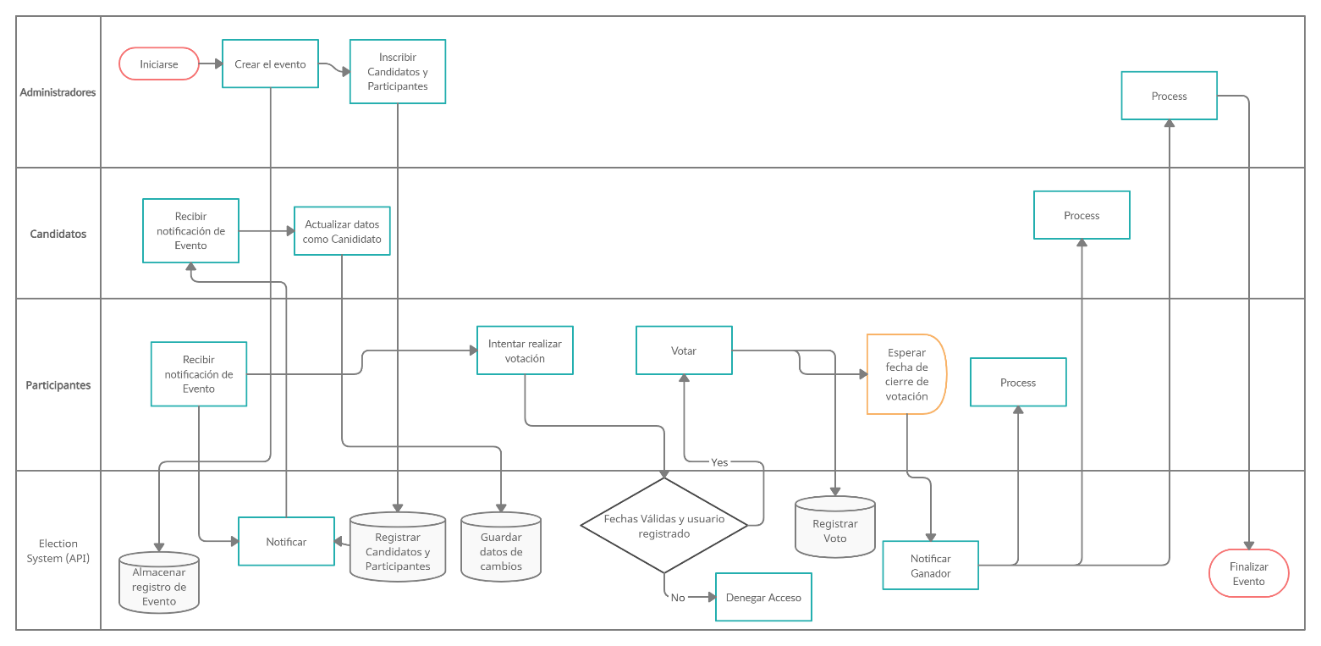
## **Diagramas de procesos**

La siguiente figura 1.1 muestra el proceso genérico actual, no automatizado con el cual se maneja las elecciones de candidatos en la mayoría de instituciones, este proceso tiene que realizar algunas validaciones que normalmente son ejecutadas por los administradores de los eventos:

1. Proceso actual, no automatizado de un evento genérico al elegir candidatos.

*Fuente: Elaboración propia.*

En la figura 1.2. se describe la forma que se va a seguir para automatizar el proceso, cabe resaltar que se toma en cuenta los mismos roles involucrados en cada proceso, pero indicando las ejecuciones automáticas que realizará el software propuesto.

1. Diagrama del proceso automatizado con el software ElectionSystem.

*Fuente: Elaboración Propia*

## **Especificación de requerimientos**

Para recopilar los requerimientos necesarios para este desarrollo se utilizó encuestas a personal de un par de instituciones que realizan este tipo de eventos; con los cuales se pudo descubrir las principales falencias que conlleva ejecutar este proceso.

Mediante las entrevistar online se propuso preguntas claves que ayudaría a especificar las posibles mejoras que debería tener este proceso al serlo llevada a un administrador de software, las encuestas se detallan más adelante en Anexos.

### Ámbito del software

El sistema propuesto tendrá el nombre de “ElectionSystemEc”, el api como tal tendrá el objetivo de ofrecer los servicios de un sistema de elecciones que pueda ser implementado y consumido desde cualquier FrontEnd.

**¿Qué hará?**

* Contendrá toda la lógica pesada del manejo, administración y ejecución de los eventos a crear.
* Permitirá que cualquier usuario pueda crear una cuenta solamente con el correo.
* Administrará fechas, reglas, validaciones para que los participantes del evento estén apegados a ellas.
* Permitirá al administrador del evento asignar mas administrador que puedan ayudarle a inscribir personas.
* Manejará el registro masivo de participantes a los eventos.
* Podrán los participantes registrarse automáticamente mediante un código único por evento.
* Notificará por correo a los participantes, candidatos asignados a un evento.
* Tendrá la funcionalidad de recuperar contraseña.

**¿Qué no hará?**

* No contará con la infraestructura para administrar eventos con una población más grande a la que encontramos en instituciones privadas o públicas.
* No permitirá eventos en los que existan listas de candidatos ya que solo permite registros únicos.
* No permitirá eventos ilimitados por administrador, cada persona solo podrá crear un evento y en caso de necesitar más, deberá contactarse con el administrador general para que le acredite la opción de más eventos.

**Metas con el sistema**

* Con el software propuesto se planea que en la actualidad los eventos realizados sean de manera virtual sin necesidad de que los participantes y candidatos tengas que asistir presencialmente a lugares de destino

### Funciones del producto

Resumen de las funciones principales que el software debe llevar a cabo. A continuación, se detalla cada uno de las Historias de Usuario que se recopilaron para el proyecto.

1. Historias de Usuario: Creación de usuarios

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Número:** 1 | **Nombre:** Creación de usuarios. |
| **Usuario:** Participante, Administrador, Candidato. | **Riesgo en Desarrollo:** Alta |
| **Prioridad en negocio:** Alta | **Iteración asignada:** 1 |
| **Descripción:** Se debe poder crear usuarios mediante el username (correo) y contraseña.   * Validar que el correo sea válido * Validar que no exista el correo registrado en la base. * Validar que la contraseña sea segura * Hashear contraseña para que el creador de software no sepa la clave original | |
| **Observación:** En caso de error mostrará un código específico que detallará el problema ocurrido en el servidor | |

1. Historias de Usuario: Permitir el acceso al sistema

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Número:** 2 | **Nombre:** Permitir el acceso al sistema. |
| **Usuario:** Participante, Administrador, Candidato. | **Riesgo en Desarrollo:** Alta |
| **Prioridad en negocio:** Alta | **Iteración asignada:** 1 |
| **Descripción:** Se debe permitir ingresar al usuario al sistema cuando ingrese su correo y contraseña correcta.   * Validar que el usuario y contraseña existan en la base de datos. | |
| **Observación:** En caso de error mostrará un código específico que detallará el problema ocurrido en el servidor | |

1. Historias de Usuario: Permitir al administrador Crear eventos

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Número:** 3 | **Nombre:** *Permitir al administrador Crear eventos.* |
| **Usuario:** Administrador | **Riesgo en Desarrollo:** Alta |
| **Prioridad en negocio:** Alta | **Iteración asignada:** 1 |
| **Descripción:** Un usuario puede funcionar como administrador de eventos ya que se le permitirá crear eventos.   * Validar que el nombre del evento no esté registrado ya como otro evento del mismo usuario. * Crear código de evento único para poder registrar participantes mediante el código. * Permitir escoger si el evento permitirá acceso libre o solamente los registrados por el administrador. * Permitir escoger si el e vento tendrá un máximo de participantes. | |
| **Observación:** En caso de error mostrará un código específico que detallará el problema ocurrido en el servidor | |

1. Historias de Usuario: Restringir la cantidad de eventos

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Número:** 4 | **Nombre:** *Restringir la cantidad de eventos.* |
| **Usuario:** Administrador | **Riesgo en Desarrollo:** Media |
| **Prioridad en negocio:** Media | **Iteración asignada:** 1 |
| **Descripción:** Al registrarse en la plataforma, cada usuario tendrá la capacidad de crear un evento, en caso de necesitar más, deberá existir un servicio con el cual se permita incrementar la cantidad de eventos permitidos.   * Crear servicio para incrementar cantidad de eventos permitidos. * Crear seguridades adicionales para el servicio | |
| **Observación:** En caso de error mostrará un código específico que detallará el problema ocurrido en el servidor | |

1. Historias de Usuario: Permitir agregar más administradores al evento

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Número:** 5 | **Nombre:** *Permitir agregar más administradores al evento* |
| **Usuario:** Administrador | **Riesgo en Desarrollo:** Alta |
| **Prioridad en negocio:** Alta | **Iteración asignada:** 1 |
| **Descripción:** Un administrador de eventos podrá agregar más usuario como administradores, los cuales tendrán las mismas autorizaciones.   * Verificar si el usuario que añade nuevos administradores es el creador del evento. | |
| **Observación:** En caso de error mostrará un código específico que detallará el problema ocurrido en el servidor | |

1. Historias de Usuario: Permitir Registrar candidatos al evento

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Número:** 6 | **Nombre:** *Permitir Registrar candidatos al evento* |
| **Usuario:** Administrador | **Riesgo en Desarrollo:** Alta |
| **Prioridad en negocio:** Alta | **Iteración asignada:** 1 |
| **Descripción:** Un administrador de eventos podrá registrar candidatos al evento, estos candidatos deben estar registrados previamente.   * Verificar si el usuario a asignar existe registrado. * Verificar si el usuario a asignar no se encuentra ya registrado como candidato en el mismo evento. | |
| **Observación:** En caso de error mostrará un código específico que detallará el problema ocurrido en el servidor | |

1. Historias de Usuario: Permitir Registrar participante al evento

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Número:** 7 | **Nombre:** *Permitir Registrar participante al evento* |
| **Usuario:** Administrador | **Riesgo en Desarrollo:** Alta |
| **Prioridad en negocio:** Alta | **Iteración asignada:** 1 |
| **Descripción:** Un administrador de eventos podrá registrar participantes al evento, estos participantes deben estar registrados previamente.   * Verificar si el usuario a asignar existe registrado. * Verificar si el usuario a asignar no se encuentra ya registrado como participante en el mismo evento. | |
| **Observación:** En caso de error mostrará un código específico que detallará el problema ocurrido en el servidor | |

1. Historias de Usuario: Permitir Registro masivo participante al evento

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Número:** 8 | **Nombre:** *Permitir Registrar participante al evento* |
| **Usuario:** Administrador | **Riesgo en Desarrollo:** Alta |
| **Prioridad en negocio:** Alta | **Iteración asignada:** 1 |
| **Descripción:** Un administrador de eventos podrá registrar participantes al evento de marena masiva mediante correos electrónicos, estos participantes deben estar registrados previamente.   * Verificar la lista de usuarios que existan registrados en la base de datos, y asignarlos al evento como participantes. * Verificar la lista de usuario que no existan registrados en la base de datos y enviarlos un correo electrónico con el código del evento. | |
| **Observación:** En caso de error mostrará un código específico que detallará el problema ocurrido en el servidor | |

1. Historias de Usuario: Permitir Registro por parte del participante

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Número:** 9 | **Nombre:** *Permitir Registro por parte del participante* |
| **Usuario:** Participante | **Riesgo en Desarrollo:** Alta |
| **Prioridad en negocio:** Alta | **Iteración asignada:** 1 |
| **Descripción:** Un participante que aún no haya sido registrado en el sistema, deberá recibir un correo electrónico por parte del sistema con el código del evento al que fue asignado, este código deberá utilizarse para registrarse en el evento.   * Cuando el administrador de eventos asigna al usuario que no existe registrado como participante, este recibirá un correo con el código único del evento. * Validar que el código enviado por el usuario sea correcto. * Validar que el administrador del evento haya registrado al usuario para participar. * Validar que el evento permita acceso libre. | |
| **Observación:** En caso de error mostrará un código específico que detallará el problema ocurrido en el servidor | |

1. Historias de Usuario: Permitir dar de baja al evento

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Número:** 10 | **Nombre:** *Permitir dar de baja al evento* |
| **Usuario:** Administrador | **Riesgo en Desarrollo:** Alta |
| **Prioridad en negocio:** Alta | **Iteración asignada:** 1 |
| **Descripción:** El creador del evento debe tener la opción para eliminar el evento creado.   * Validar que solamente el creador del evento pueda eliminar el evento. * Validar que el evento no haya empezado para poder eliminar. | |
| **Observación:** En caso de error mostrará un código específico que detallará el problema ocurrido en el servidor | |

1. Historias de Usuario: Permitir dar de baja candidato por evento

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Número:** 10 | **Nombre:** *Permitir dar de baja candidato por evento* |
| **Usuario:** Administrador | **Riesgo en Desarrollo:** Alta |
| **Prioridad en negocio:** Alta | **Iteración asignada:** 1 |
| **Descripción:** Los administradores del evento deberán poder eliminar los candidatos registrados al evento.   * Validar que el candidato a eliminar esté registrado * Validar que el evento no haya empezado. * Validar que el usuario que elimina sea administrador del evento. | |
| **Observación:** En caso de error mostrará un código específico que detallará el problema ocurrido en el servidor | |

1. Historias de Usuario: Permitir dar de baja participante por evento

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Número:** 11 | **Nombre:** *Permitir dar de baja participante por evento* |
| **Usuario:** Administrador | **Riesgo en Desarrollo:** Alta |
| **Prioridad en negocio:** Alta | **Iteración asignada:** 1 |
| **Descripción:** Los administradores del evento deberán poder eliminar los participantes registrados al evento.   * Validar que el participante a eliminar esté registrado * Validar que el participante no haya empezado. * Validar que el usuario que elimina sea administrador del evento. | |
| **Observación:** En caso de error mostrará un código específico que detallará el problema ocurrido en el servidor | |

1. Historias de Usuario: Permitir actualizar Datos del evento

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Número:** 12 | **Nombre:** *Permitir actualizar Datos del evento* |
| **Usuario:** Administrador | **Riesgo en Desarrollo:** Alta |
| **Prioridad en negocio:** Alta | **Iteración asignada:** 1 |
| **Descripción:** Los administradores del evento podrán actualizar los datos del evento.   * Validar que el participante no haya empezado. * Validar que el usuario que actualiza sea administrador del evento. | |
| **Observación:** En caso de error mostrará un código específico que detallará el problema ocurrido en el servidor | |

1. Historias de Usuario: Permitir Actualizar datos del usuario

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Número:** 13 | **Nombre:** *Permitir Actualizar datos del usuario* |
| **Usuario:** Administrador, Participante, Candidato | **Riesgo en Desarrollo:** Alta |
| **Prioridad en negocio:** Alta | **Iteración asignada:** 1 |
| **Descripción:** Los usuarios registrados en el sistema deberán poder actualizar sus datos personales.   * Validar que el usuario tenga un token valido para realizar esta acción. * Validar que no pueda cambiar el correo electrónico registrado. | |
| **Observación:** En caso de error mostrará un código específico que detallará el problema ocurrido en el servidor | |

1. Historias de Usuario: Permitir Actualizar datos del candidato

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Número:** 14 | **Nombre:** *Permitir Actualizar datos del candidato* |
| **Usuario:** Candidato | **Riesgo en Desarrollo:** Alta |
| **Prioridad en negocio:** Alta | **Iteración asignada:** 1 |
| **Descripción:** Los candidatos registrados en el evento deberán poder actualizar su información.   * Validar el candidato esté registrado en el evento. * Validar que le evento no haya empezado. | |
| **Observación:** En caso de error mostrará un código específico que detallará el problema ocurrido en el servidor | |

1. Historias de Usuario: Permitir a los participantes votar

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Número:** 15 | **Nombre:** *Permitir a los participantes vota.* |
| **Usuario:** Participantes | **Riesgo en Desarrollo:** Alta |
| **Prioridad en negocio:** Alta | **Iteración asignada:** 1 |
| **Descripción:** Los participantes registrados en el evento deberán poder votar por el candidato que desean.   * Validar que el participante esté registrado en el evento. * Validar que el participante no haya realizado ya su voto. * Validar las fechas para realizar votación. | |
| **Observación:** En caso de error mostrará un código específico que detallará el problema ocurrido en el servidor | |

1. Historias de Usuario: Los registros y votaciones deberán tener fechas de control.

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Número:** 16 | **Nombre:** *Los registros y votaciones deberán tener fechas de control.* |
| **Usuario:** Administradores, Participantes, Candidatos | **Riesgo en Desarrollo:** Alta |
| **Prioridad en negocio:** Alta | **Iteración asignada:** 1 |
| **Descripción:** Se debe controlar fechas máximas para inscribir candidatos, registrar participantes; fechas mínimas y máximas para realizar votaciones.   * Registrar fechas requeridas al crear eventos. | |
| **Observación:** En caso de error mostrará un código específico que detallará el problema ocurrido en el servidor | |

1. Historias de Usuario: Dar seguridad al login mediante tokens

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Número:** 17 | **Nombre:** *Dar seguridad al login mediante tokens* |
| **Usuario:** Usuarios | **Riesgo en Desarrollo:** Alta |
| **Prioridad en negocio:** Alta | **Iteración asignada:** 1 |
| **Descripción:** Al ingresar los datos correctos de logín, el servicio deberá devolver un token con el cual podrá utilizar todos los demás servicios.   * Controlar todos los servicios que necesiten seguridad. * Devolver el token al ingresar datos correctos en el login. * Añadir en el token Claims para contexto en toda la app. | |
| **Observación:** En caso de error mostrará un código específico que detallará el problema ocurrido en el servidor | |

1. Historias de Usuario: Permitir subir imágenes al servidor

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Número:** 18 | **Nombre:** *Permitir subir imágenes al servidor* |
| **Usuario:** Usuarios | **Riesgo en Desarrollo:** Alta |
| **Prioridad en negocio:** Alta | **Iteración asignada:** 1 |
| **Descripción:** Los eventos y los candidatos deben permitir subir imágenes que después serán mostradas al usuario.   * Crear módulo para guardar imágenes en el servidor. * Recuperar links de imágenes en consulta. * Evitar redundancia en imágenes. | |
| **Observación:** En caso de error mostrará un código específico que detallará el problema ocurrido en el servidor | |

1. Historias de Usuario: Notificar por correo diferentes acciones.

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Número:** 19 | **Nombre:** *Notificar por correo diferentes acciones.* |
| **Usuario:** Usuarios | **Riesgo en Desarrollo:** Alta |
| **Prioridad en negocio:** Alta | **Iteración asignada:** 1 |
| **Descripción:** Los servicios de: Crear usuario, registrar candidato, registrar participante, olvidó contraseña deben enviar correos a los emails destinos.   * Validar que existan correos registrados | |
| **Observación:** En caso de error mostrará un código específico que detallará el problema ocurrido en el servidor | |

1. Historias de Usuario: Implementar servicio para olvidó contraseña

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Número:** 20 | **Nombre:** *Implementar servicio para olvidó contraseña.* |
| **Usuario:** Usuarios | **Riesgo en Desarrollo:** Alta |
| **Prioridad en negocio:** Alta | **Iteración asignada:** 1 |
| **Descripción:** Si el usuario olvidó su contraseña deberá haber un servicio que le permita generar una temporal para cambiar la contraseña.   * Validar que exista el correo electrónico. | |
| **Observación:** En caso de error mostrará un código específico que detallará el problema ocurrido en el servidor | |

### Características de los usuarios del sistema

La API tendrá 3 diferentes roles que un usuario puede llevar:

Administrador: Un usuario registrado tiene la opción de crear eventos, al hacerlo se vuelve el administrador de dicho evento y tiene las autorizaciones debidas para poder realizar acciones como Actualizar el evento, crear nuevos administradores del evento, eliminar evento, etc.

Candidato: Un usuario registrado puede ser asignado como candidato para un evento, al hacerlo tendrá la opción de actualizar la información como candidato, subir imágenes, etc.

Participante: Un usuario registrado tiene la opción de ser asignado como participante de un evento, lo cual le permitirá realiza el proceso de votación.

En la siguiente tabla se muestra un ejemplo de la forma de presentar las características de los usuarios con respecto a sus perfiles.

1. Perfiles de usuario

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre de Usuario** | **Tipo de Usuario** | **Área Funcional** | **Actividad** |
| **Usuario** | Principal, los demás roles pueden hacer lo mismo | Administración | Registrase  Actualizar Información  Recuperar Contraseña |
| **Administrador.** | Administrador | Administración | Crear eventos  Actualizar Eventos  Dar de baja Eventos  Registrar participantes  Registrar participantes masivamente  Dar de baja Participante.  Registrar candidatos.  Dar de baja Candidatos.  Consultar Candidatos  Consultar Eventos  Consultar Participantes  Ver resultados de evento |
| **Candidato** | Candidato | Administración | Actualizar información de candidato.  Ver resultados de evento |
| **Participante** | Participante | Administración | Ver participantes del evento  Ver candidatos del evento  Ver resultados de evento |

### Restricciones

A continuación, se detalla las restricciones que utilizará la aplicación para poder funcionar:

**Servidor**

* Para el ambiente de producción necesitaremos una máquina que soporte despliegues en IIS con SDK Runtime .net core 5.0, ya que todo el proyecto fue realizado con el lenguaje de programación C#.
* La forma en la que almacenamos la información es mediante MS-SQL 2017 -2019, utilizamos esta tecnología debido a que es parte de la familia de Microsft y se acopla e integra perfectamente con el lenguaje de programación C# que también pertenece a Microsoft.
* El servicio de correos electrónicos está a cargo de la aplicación SendGrid, la cual es un servicio que se encarga de administrar todas las peticiones de correo enviadas, además nos permite visualizar estadísticas del número de veces que se envía correos, las horas, etc.

### Requisitos

1. Requerimientos Funcionales

|  |  |
| --- | --- |
| **Requerimientos funcionales del sistema** | |
| **Número de Requerimiento** | **Descripción** |
| **RF: 01** | Permitir a las personas registrase en la aplicación. |
| **RF: 02** | Permitir a los usuarios ya registrados, recuperar su contraseña. |
| **RF: 03** | Permitir a los usuarios actualizar información. |
| **RF: 04** | Permitir a los usuarios crear, modificar, eliminar eventos. |
| **RF: 05** | Permitir a los administradores crear, eliminar, consultar candidatos. |
| **RF: 06** | Permitir a los administradores crear, eliminar, consultar participantes. |
| **RF: 07** | Permitir a los candidatos actualizar su información |
| **RF: 08** | Permitir a los candidatos consultar su información. |
| **RF: 09** | Permitir a los participantes votar en evento. |
| **RF: 10** | Permitir los usuarios consultar resultado de evento |

#### No funcionales.

1. Requerimientos no Funcionales

|  |  |  |
| --- | --- | --- |
| **Requerimientos no funcionales del sistema** | | |
| **Categoría** | **Número** | **Descripción** |
| **Usabilidad** | RNF: 01 | El api deberá estar disponible las 24 horas del día para ser consumida por los implementadores y usada por el usuario final. |
| RNF: 02 | El api constará con una guía en Swagger para explicar los métodos existentes adicional de los parámetros necesarios por cada método. |
| **Portabilidad** | RNF: 03 | Ya que es un api, podrá se consumida desde cualquier frente que quiera implementarlo, ya sea dispositivos móviles, páginas webs o aplicaciones de escritorio. |
| **Seguridad** | RNF: 04 | La información valiosa para el usuario como su contraseña será aplicada hasheo para evitar que los administradores de base de datos conozcan su acceso. |
| RNF: 05 | Toda la aplicación será manejada mediante un contexto envía en un token con protección, de esta manera evitaremos cualquier intento de violación al login o servicios. |

# 

# CAPÍTULO 2. RESULTADOS

## Diseño general

Para la construcción de toda el API utilizaremos el sistema Scrum detallado a continuación:

1. Tabla de Scrum

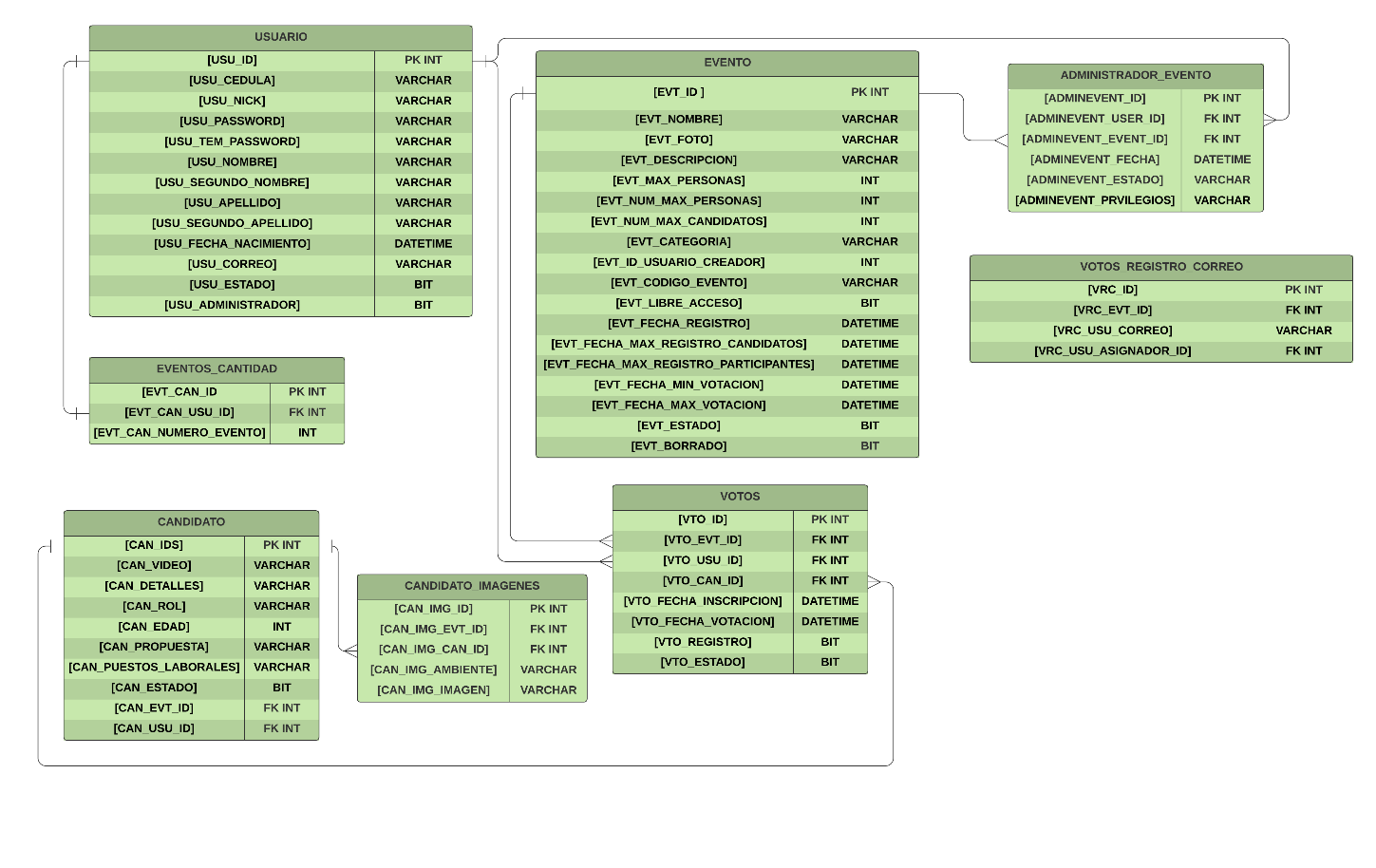
|  |  |  |
| --- | --- | --- |
| **SCRUM** | | |
| **Sprint** | **Objetivo** | **Entregable** |
| **1** | Crear estructura del proyecto | Estructura base con todo el modelo que se usará en el resto de proyecto |
| **2** | Cruds de tablas Usuario, Administrador | Servicios que permita realizar crud de Usuarios y Administradores |
| **3** | Cruds de tablas Eventos, Votos, Candidatos | Servicios que permita realizar crud de Eventos, Votos, Candidatos |
| **4** | Validaciones | Servicios con sus respectivas validaciones al crear o actualizar datos. |
| **5** | Notificaciones y Correos | Servicios integrados la notificación por correo en eventos determinados. |
| **6** | Test y revisión de Código | Código probado, con test unitarios y escaneado con SonarQube para corrección de vulnerabilidades. |

## Esquema de la base de datos (SGBDD)

El esquema de base de datos utilizado para manejar todos los requisitos necesarios, requiere unas 8 tablas las cuales son:

* USUARIO
* EVENTO
* ADMINISTRADOR\_EVENTO
* EVENTO\_CANTIDAD
* CANDIDATO
* CANDIDATO\_IMAGENES
* VOTOS
* VOTOS\_REGISTRO\_CORREO

1. Tabla de base de datos principal



*Fuente: Elaboración Propia*

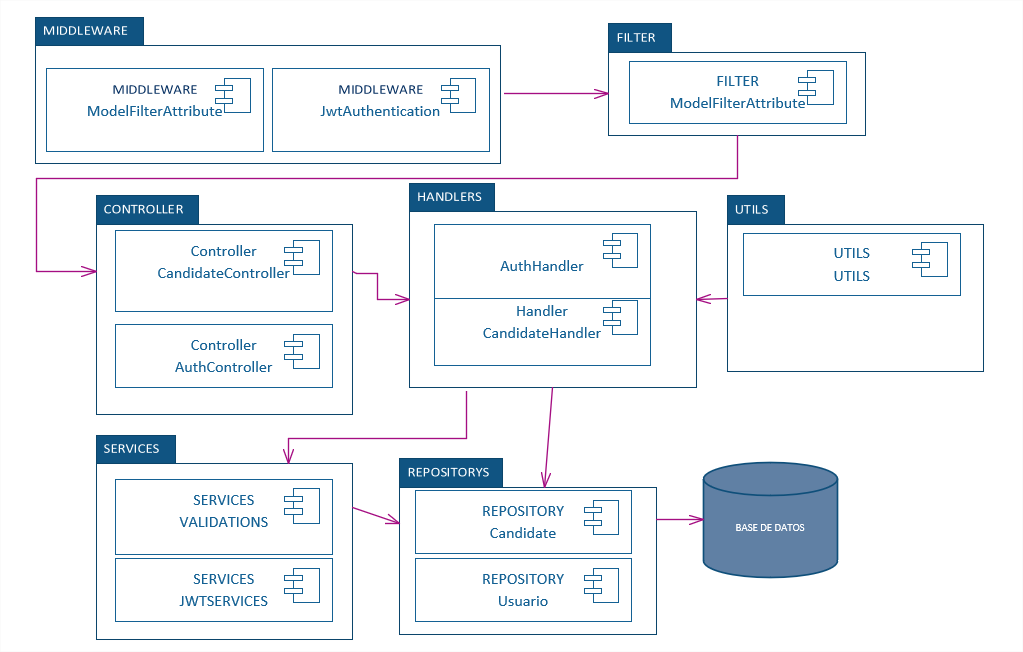
## Diagrama de la arquitectura del sistema

El Api para el aplicativo Election System, tendrá el siguiente esquema:

1. Arquitectura del Sistema

|  |  |
| --- | --- |
| **Arquitectura del sistema** | |
| **Componente** | **Descripción** |
| Middleware | El componente Middleware se encargará de recibir los request que ingresen en el Api, luego podrá realizar validaciones generales como son: validar que el JWT sea verdadero.  También tendremos un middleware general que capturará cualquier excepción que pueda existir en la aplicación. |
| Filter | Filter es una capa previa al controlador, en la cual los request serán inyectados el contexto del usuario para poder utilizar en todos los servicios. |
| Controller | Luego de pasar por los filtros, existirá la capa de controladores la cual se encargará de recibir el request enviado, con su respectivo modelo de datos. |
| Handler | Cada request tendrá su manejador (Handler) en la cual se aplicará toda la lógica del negocio. |
| Utils | La capa de utilitario que podemos añadir cualquier componente que utilizaremos en toda la app. |
| Services | La capa de servicios se conectará con la capa Handlers y Repository; en esta capa podremos implementar lógica que se utiliza frecuentemente en los manejadores, y de esta manera reutilizar código. |
| Repository | La capa de repositorio tendrá la persistencia de datos y el enlace con cada una de las tablas de la base de datos. |
| DataBase | La base de datos como tal, contendrá todas las tablas que utilizaremos en la aplicación. |

1. Arquitectura del Sistema

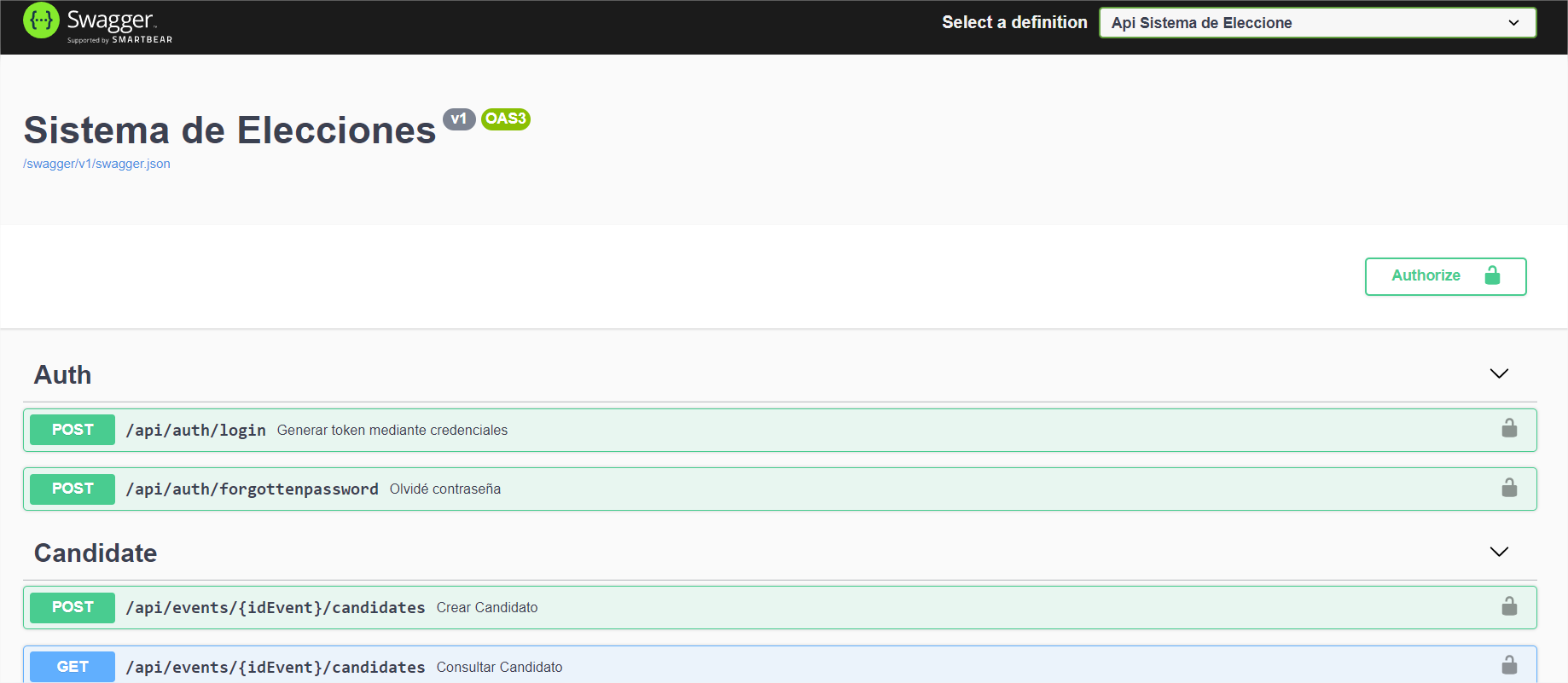


*Fuente: Elaboración Propia*

### Diseño de interfaces

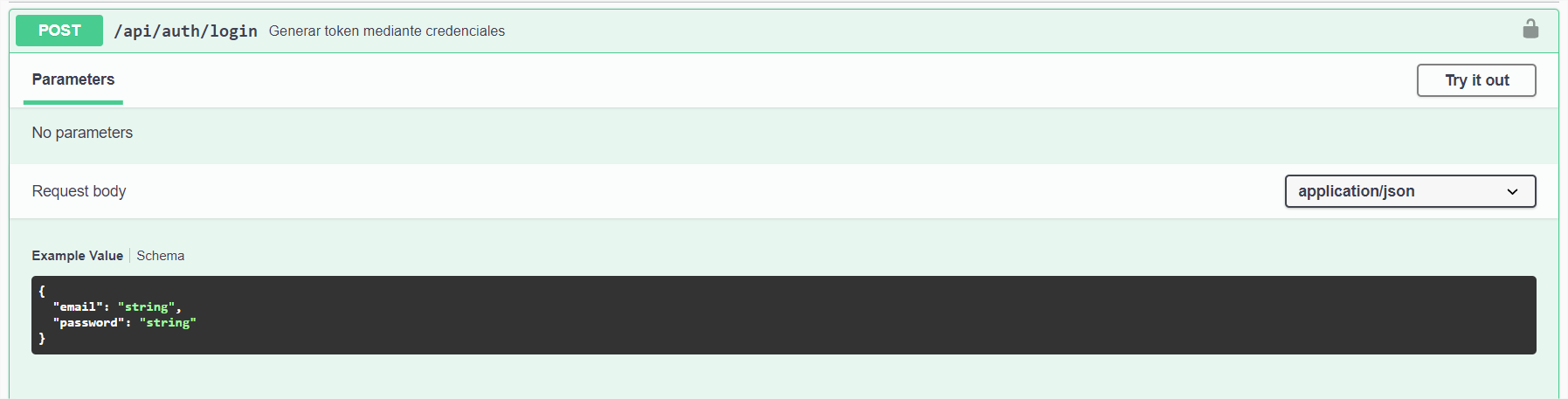
Debido a que el software a implementar es un API, no posee interfaces ni diseños previstos ya que todos esos componentes los debe implementar el desarrollador que vaya a utilizar los servicios entregados y visualizados en Swagger.

1. Vista General de Pantalla Swagger



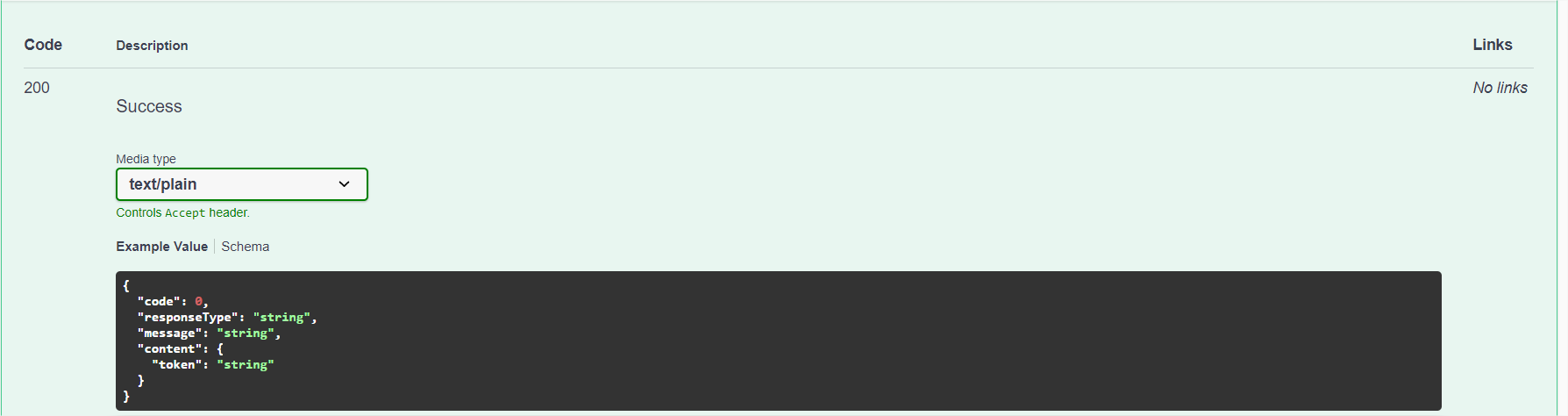
*Fuente: Elaboración Propia*

1. Vista de muestra del método login (swagger)



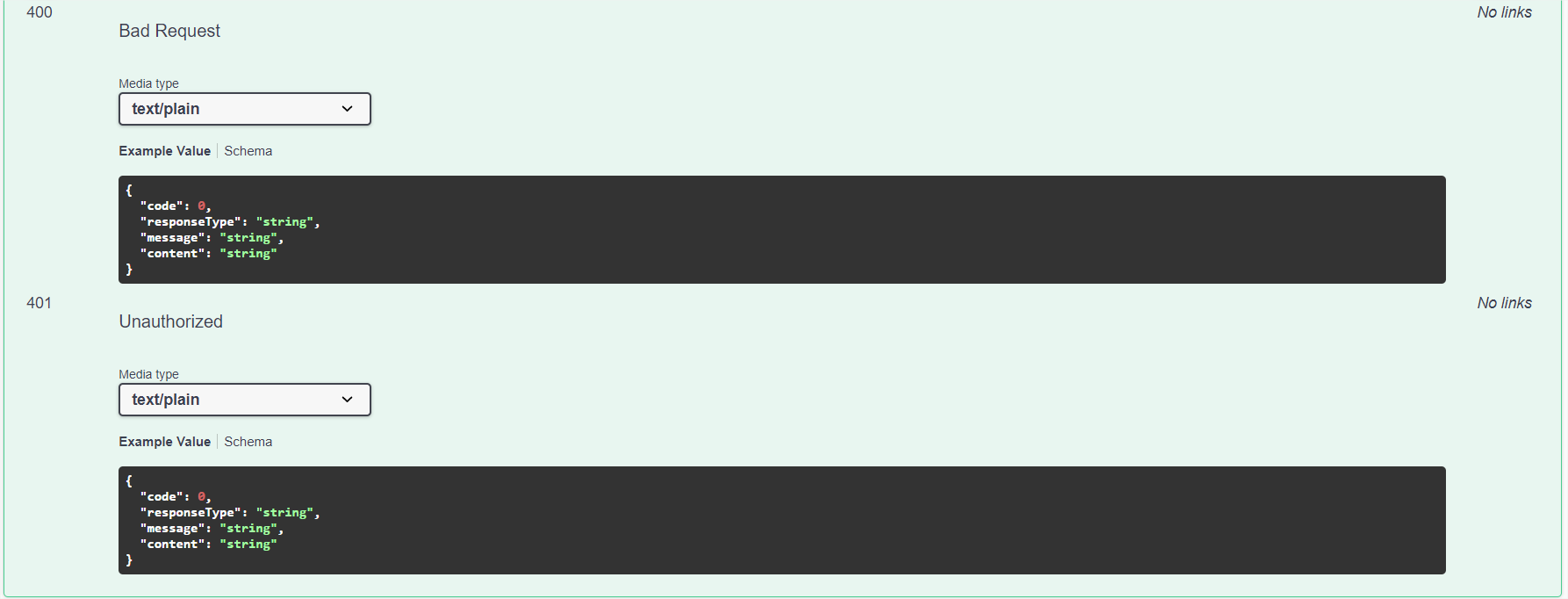
*Fuente: Elaboración Propia*

1. Vista de muestra del método login en caso de ser correcta la petición (swagger)



*Fuente: Elaboración Propia*

1. Vista de muestra del método login en caso de ser errónea la petición (swagger)



*Fuente: Elaboración Propia*

## Estándares de programación utilizados

A continuación, se detalla los estándares de programación utilizados para la creación de todos los componentes utilizados en el API.

1. Estándares de programación utilizados

|  |  |  |
| --- | --- | --- |
| **Estándares de programación** | |  |
| **Estandar** | **Descripción** | **Ejemplo Usado** |
| Camelcase | CamelCase es un tipo de escritura de programación. La notación Camel Case combina las palabras directamente, sin usar ningún símbolo, estableciendo que la primera letra de cada palabra esté en mayúscula a excepción de la primera palabra, estando el resto de letras en minúsculas. Este tipo de notación está muy extendida, siendo su uso muy común tanto en la declaración de variables como en el nombre de funciones y métodos.(Tipos de notación: Camel Case, Pascal Case, Snake Case y Kebab Case | Neoguias n.d.) | Nombre de Clases: ApiControllerBase  Nombre de Atributos privados: \_userMessages.  Nombre de métodos: Success() |
| Entity Framework | Entity Framework Core es una tecnología de acceso a datos para .NET Core y .NET Framework. Es multiplataforma y de código abierto desarrollado por Microsoft con aportes de la comunidad. Propiamente dicho es un asignador objeto relacional o ORM por sus siglas en inglés. Su función principal es servir como interprete entre dos tecnologías fundamentadas en distintos principios por un lado la programación orientada a objetos y por el otro las bases de datos relacionales y no relacionales.(Introdución a Entity Framework Core | ASP.NET Core Master n.d.) | [Key, Column("CAN\_IMG\_ID")]  DatabaseGenerated(DatabaseGeneratedOption.Identity)]  public int Id { get; set; } |
| Patrón Repository | El patrón repositorio o repository pattern está íntimamente relacionado con el acceso a datos y nos permite tener una abstracción de la implementación de acceso a datos en nuestras aplicaciones, de modo que nuestra lógica de negocio no conozca ni esté acoplada a la fuente de datos. En pocas palabras esto quiere decir que el repositorio actúa como un intermediario entre nuestra lógica de negocio y nuestra lógica de acceso a datos para que se centralice en un solo punto, y de esta forma se logre evitar redundancia de código.(El Tavo = { c#, asp.net, MVC, WCF, y más}: [Patrones] Implementando patrón repositorio - Repository pattern en C# Parte I n.d.) |  |
| Seguridad JWT | JWT (JSON Web Token) es un estándar qué está dentro del documento RFC 7519. En el mismo se define un mecanismo para poder propagar entre dos partes, y de forma segura, la identidad de un determinado usuario, además con una serie de claims o privilegios. Estos privilegios están codificados en objetos de tipo JSON, que se incrustan dentro de del payload o cuerpo de un mensaje que va firmado digitalmente.(Qué es Json Web Token y cómo funciona | OpenWebinars n.d.) | Ejemplo de token devuelto en petición: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJOYW1lIjoiUHJ1ZWJhczIiLCJFbWFpbCI6InBydWViYXMyQGdtYWlsLmNvbSIsIklkIjoiMiIsIm5iZiI6MTYyMjQzMjk3OSwiZXhwIjoxNjIyNDM0Nzc5LCJpYXQiOjE2MjI0MzI5Nzl9.vXS--U38vISXE3280BJjocUnwIN\_uACJz9lIMOFX8tU.  Este token no se puede descifrar sin la clave de acceso |
| Identificadores de tipo de clases | Los identificadores de clase son una nomenclatura o buena práctica para poder distinguir el tipo de objeto a crear, por ejemplo, cuando creamos una interfaz, es recomendable colocar la letra ‘I’ por delante para identificar el tipo de objeto. | Controladores: AuthController  Interfaces: IUserRepository  Clase Base: EventBase |
| Inyección de dependencias | La inyección de dependencias (DI, por sus siglas en inglés) es un patrón usado en el diseño orientado a objetos de una aplicación. Es parte de uno de los cinco principios de diseño de clases conocido como S.O.L.I.D. Como todo patrón de diseño, DI tiene como finalidad solucionar un problema común que los programadores encuentran en la construcción de aplicaciones. Este es, mantener los componentes o capas de una aplicación lo más desacopladas posible. (¿Qué es la inyección de dependencias, para qué sirve y qué significan los tiempos de vida en su implementación? - Baufest n.d.) |  |
| Loggers | Logging proviene del término en inglés “logˮ y, en este contexto, se refiere a un protocolo. Al igual que un libro de registro, contiene todos los registros importantes del historial de eventos. Dependiendo del tipo de seguimiento que queramos hacer, solo se registran ciertas acciones o eventos de un proceso o, por el contrario, se comprueban todas las acciones.(El módulo logging de Python: así funciona logging to file - IONOS n.d.) | Utilizamos logs en la información principal a registrar como los Request, Errores, etc.  [22:28:36][WRN][Dach.ElectionSystem.BusinessLogic.User.UserCreateHandler]:No se pudo Envíar correo |
| UnitOfwork | Unidad de trabajo (unit of work) es el patrón que permite manejar transacciones durante la manipulación de datos utilizando el patrón repositorio. Mantiene una lista de los objetos afectados por una transacción de negocio y coordina la escritura de los cambios y la resolución de problemas de concurrencia.(Data access patterns: Unit of Work & Repository - Kabel n.d.) | Ejemplo utilizando UnitOfWork:  await \_electionUnitOfWork.BeginTransactionAsync().ConfigureAwait(false);  var hasCreate = await \_electionUnitOfWork.GetEventRepository().CreateAsync(newEvent); |
| Patron Mediator | El patrón mediador define un objeto que encapsula cómo un conjunto de objetos interactúa. Este patrón de diseño está considerado como un patrón de comportamiento debido al hecho de que puede alterar el comportamiento del programa en ejecución. Habitualmente un programa está compuesto de un número de clases (muchas veces elevado). La lógica y computación es distribuida entre esas clases. Sin embargo, cuantas más clases son desarrolladas en un programa, especialmente durante mantenimiento y/o refactorización, el problema de comunicación entre estas clases quizás llegue a ser más complejo. (Mediator (patrón de diseño) - Wikipedia, la enciclopedia libre n.d.) | Ejemplo Utilizando Mediator:  Success(await \_mediator.Send(requestLogin)); |

## Pruebas

**Pruebas de Estrés**

Las pruebas de estrés sirven para poder medir la capacidad que tienen los servidores web para responder a las peticiones enviadas por los usuarios, en el caso específico de APIs se dispone de “disparadores” que crean bucles de peticiones para someter al software a un uso máximo de sus capacidades y poder medir su rendimiento adecuado en un ambiente de desarrollo.

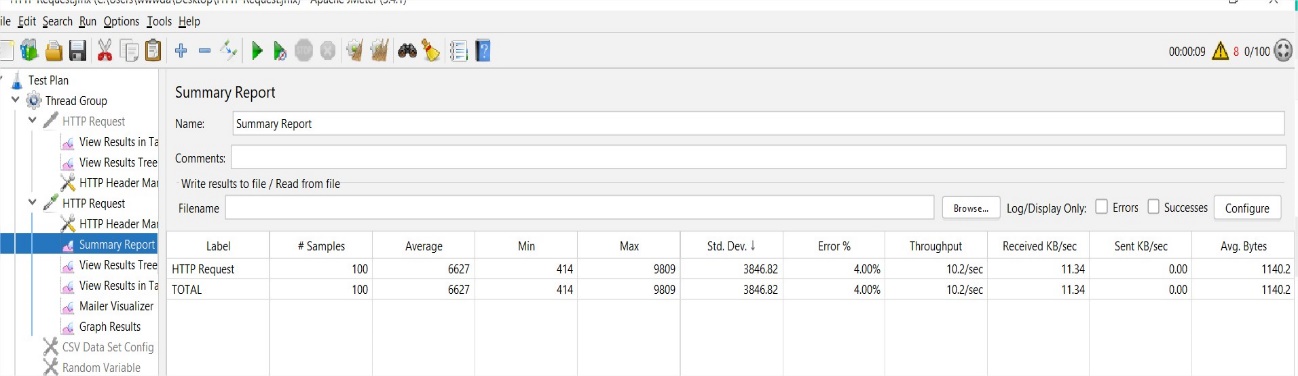
JMeter, es una herramienta de pruebas de carga fuerte y con características pesadas. Este programa puede cargar recursos de prueba tanto dinámicos como estáticos al generar usuarios concurrentes para crear un entorno de prueba de la vida real para reconocer los cuellos de botella en cualquier aplicación web o móvil. Cuando hablamos de recursos estáticos y dinámicos, los recursos estáticos podrían ser lenguajes y marcos como JavaScript o HTML. Los recursos dinámicos podrían ser marcos e idiomas como AJAX, Flex.(5 ejemplos de pruebas de carga de JMeter - LoadView n.d.)

Para estas pruebas se utilizó un ambiente de desarrollo real montando en la web con características básicas que permitirá calcular el rendimiento en caso de utilizar un entorno de producción más fuerte:

1. Pruebas de estrés Jmeter, servicio Post “Create User” - www.electionsystem.ml/ /api/users.

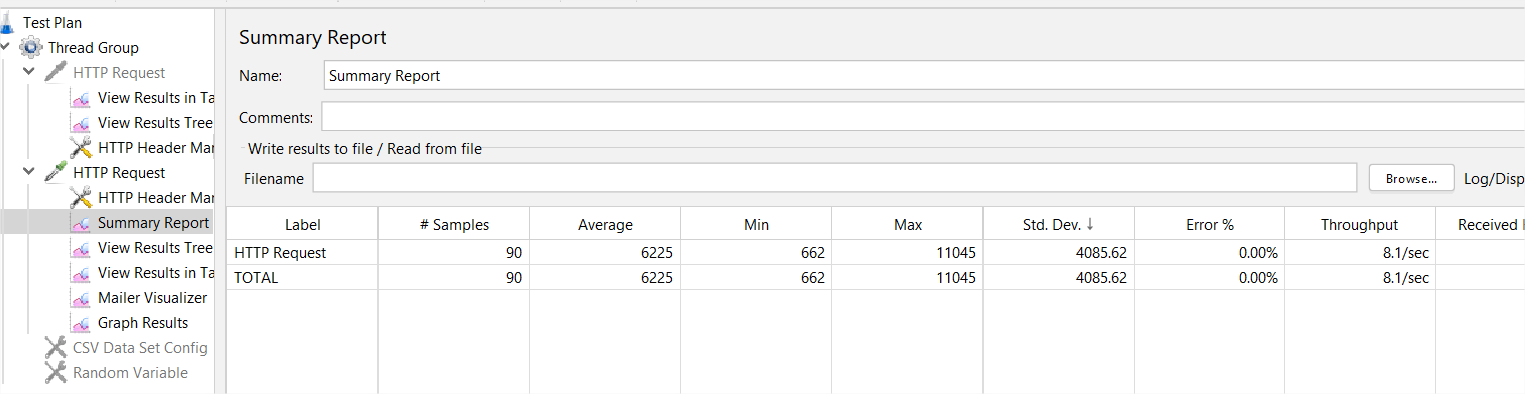
En la prueba realizada el servicio Create User, se encontró que el servidor resistía una carga de 90 Usuarios, como se puede ver una la figura 10, ya que, al someterlo a una carga con 100 Usuarios, como se puede ver en la figura 9, existió peticiones que no se completaron y eso puede llevar a inconsistencias de datos o simplemente mala experiencia de usuario.

1. Prueba de Carga al Servicio “Create User” con 100 Usuarios



*Fuente: Elaboración Propia*

1. Prueba de Carga al servicio “Create User” con 90 Usuario

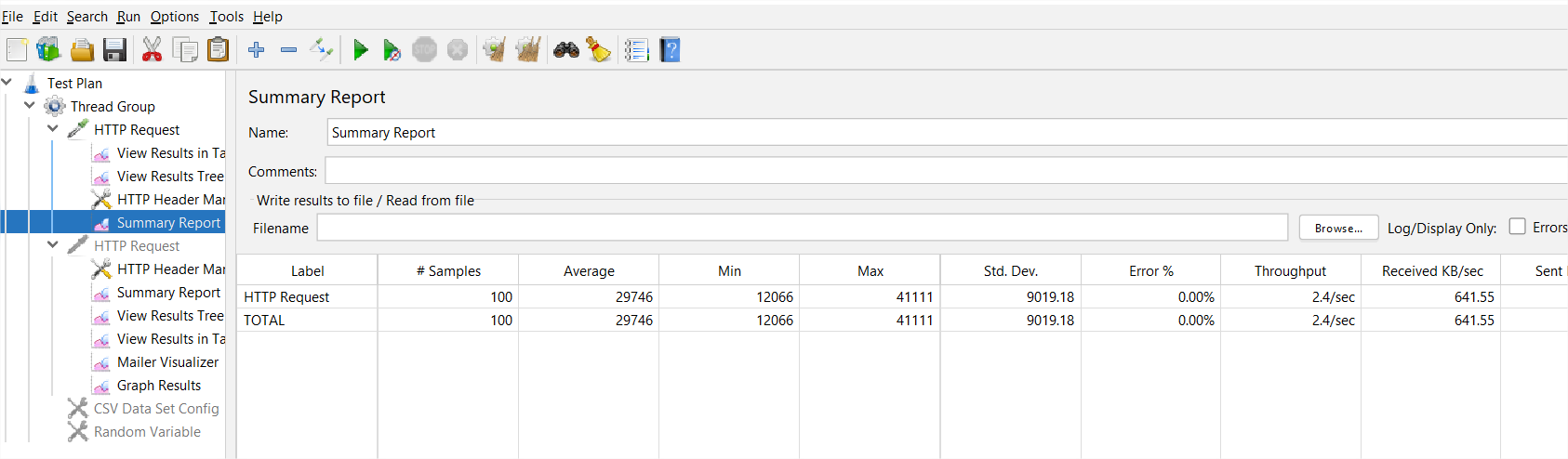


*Fuente: Elaboración Propia*

1. Pruebas de estrés Jmeter, servicio Get “Get User” - www.electionsystem.ml/ /api/users.

En el servicio get se pudo comprobar que todas las ejecuciones fueron completas, pero el tiempo de respuesta incrementaba desde los 3 segundos hasta los 29 segundos en una prueba de consulta de 1000 registros.

1. Prueba de carga al servicio “Get User” con 100 Usuarios y 1000 consultas



*Fuente: Elaboración Propia*

**Análisis de código SonarQube**

Para poder asegurar que el Api se encuentre en un estado optimo de codificación, se utilizará el escáner de SonarQube.

SonarQube es una herramienta que permite realizar un análisis estático de código. Lo que hace es identificar los puntos susceptibles de mejora, que facilitarán la obtención de métricas necesarias para la optimización del código.

Una de las ventajas de SonarQube es que soporta más de 20 lenguajes de programación, y cada uno de ellos cuenta con una serie de reglas que permiten detectar problemas generales o específicos de un lenguaje particular. (SonarQube: Una herramienta útil para verificar la calidad del código - n.d.)

1. El

## Implementación

Colocar lo que se requiere si se implementaría la solución.

### Requerimientos de hardware y software

Describir los requerimientos de Hardware y Software que se requieren para la implementación, tomar en cuenta tanto para el servidor como para los usuarios finales.

#### Software para servidor

* SGBD con su respectiva versión, esto es un ejemplo.
* Servidor Web, esto es un ejemplo.
* Servidor de aplicaciones, esto es un ejemplo.
* Aplicaciones de terceros (Ms-office, Adobe, WinRar), etc., esto es un ejemplo.

#### Software para usuario

* Lector de PDF, esto es un ejemplo.
* Navegador web (Chrome, Firefox, I-Explorer y sus versiones), esto es un ejemplo.

#### Hardware para servidor

* RAM al menos de 8Gb, esto es un ejemplo.
* Procesador Core I7, esto es un ejemplo.
* Tarjeta de Red 1Gbps, esto es un ejemplo.

#### Hardware para usuario

RAM al menos de 2Gb, esto es un ejemplo.

# CONCLUSIONES

Valoración general del trabajo presentado, destacar el aporte y las generalizaciones que pueden hacerse de todo el proceso investigativo. Es importante ajustarse en las Conclusiones a los resultados obtenidos en cada uno de los Capítulos y no hacer referencias a aspectos que necesitan continuar siendo estudiados y que no quedaron resueltos por salirse del Campo de Acción de la Investigación. Debe existir al menos una conclusión por cada objetivo específico planteado.

* Conclusión 1
* Conclusión 2
* ….

# RECOMENDACIONES

Deben ser aquellas que no están al alcance del Autor(es) en el momento de culminación del trabajo, pero que pueden obtenerse en un periodo de post- grado, o que pueden ser resueltas en otras instancias por su factibilidad y beneficio para la misma. También deben estar en correspondencia con el campo de acción de la investigación o marco de desarrollo de la misma. Un error muy frecuente es incluir en conclusiones aspectos que se refieren a recomendaciones. Recomendar la divulgación de los resultados. El número de recomendaciones no debe exceder el de las conclusiones

* Recomendación 1
* Recomendación 2
* ….

# REFERENCIAS BIBLIOGRÁFICAS

“¿Qué Es La Inyección de Dependencias, Para Qué Sirve y Qué Significan Los Tiempos de Vida En Su Implementación? - Baufest.” http://www.tss.com.pe/blog/que-es-la-inyeccion-de-dependencias-para-que-sirve-y-que-significan-los-tiempos-de-vida-en-su-implementacion (May 20, 2021).

“5 Ejemplos de Pruebas de Carga de JMeter - LoadView.” https://www.loadview-testing.com/es/blog/5-ejemplos-de-pruebas-de-carga-de-jmeter/ (May 30, 2021).

“Data Access Patterns: Unit of Work & Repository - Kabel.” https://www.kabel.es/data-access-patterns-unit-of-work-repository/ (May 20, 2021).

“El Módulo Logging de Python: Así Funciona Logging to File - IONOS.” https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/logging-de-python/ (May 20, 2021).

“El Tavo = { C#, Asp.Net, MVC, WCF, y Más}: [Patrones] Implementando Patrón Repositorio - Repository Pattern En C# Parte I.” http://www.eltavo.net/2014/08/patrones-implementando-patron.html (May 20, 2021).

“Introdución a Entity Framework Core | ASP.NET Core Master.” https://aspnetcoremaster.com/c%23/entityframeworkcore/dotnet/ef/2018/08/07/introduccion-entityframeworkcore.html (May 19, 2021).

“Mediator (Patrón de Diseño) - Wikipedia, La Enciclopedia Libre.” https://es.wikipedia.org/wiki/Mediator\_(patrón\_de\_diseño) (May 20, 2021).

“Qué Es Json Web Token y Cómo Funciona | OpenWebinars.” https://openwebinars.net/blog/que-es-json-web-token-y-como-funciona/ (May 20, 2021).

“SonarQube: Una Herramienta Útil Para Verificar La Calidad Del Código -.” https://castor.com.co/sonarqube-una-herramienta-util-para-verificar-la-calidad-del-codigo/ (May 31, 2021).

“Tipos de Notación: Camel Case, Pascal Case, Snake Case y Kebab Case | Neoguias.” https://www.neoguias.com/tipos-notacion-nombres/ (May 19, 2021).

# ANEXOS