UNIVERSIDAD POLITECNICA INTERNACIONAL

ESCUELA DE INGENIERIA

BACHILLERATO EN INGENIERIA SISTEMAS

Asignatura:

Programacion III

Proyecto Final

Tema:

Sistema de Votación

Estudiantes:

Mercy Natalia Tobal

Jonathan Morales

Lindsey Mendieta Chavez

Profesor:

Alexander Benjamin Curling

SEDE UNIVERSITARIA

Heredia

SEGUNDO CUATRIMESTRE 2024

Tabla de Contenidos

[1 Índice de Ilustraciones 3](#_Toc131013710)

[2 índice de Tablas 4](#_Toc131013711)

[3 Resumen Ejecutivo 5](#_Toc131013712)

[4 Introducción 6](#_Toc131013713)

[4.1 Objetivos generales 6](#_Toc131013714)

[4.2 Objetivos Específicos 6](#_Toc131013715)

[5 Marco teórico 7](#_Toc131013716)

[6 Marco Metodológico 8](#_Toc131013717)

[7 Resultados de la Experimentación e Investigación 9](#_Toc131013718)

[8 Análisis de Resultados 10](#_Toc131013719)

[9 Conclusiones 11](#_Toc131013720)

[10 Recomendaciones 12](#_Toc131013721)

[11 Bibliografías 13](#_Toc131013722)

[12 Apéndices y Anexos 14](#_Toc131013723)

# Índice de Ilustraciones

**No se encuentran elementos de tabla de ilustraciones.**

# índice de Tablas

**No se encuentran elementos de tabla de ilustraciones.**

# Resumen Ejecutivo

Este proyecto de programación se enfoca en el desarrollo de un sistema de votación utilizando el patrón de diseño MVC (Modelo-Vista-Controlador), implementado con tecnologías como C#, HTML, CSS, SQL, y Bootstrap. El objetivo principal es crear una plataforma de votación que sea intuitiva, accesible y amigable para los usuarios, facilitando el proceso de votación de manera eficiente y segura.

## Arquitectura y Tecnología

El sistema está diseñado siguiendo el patrón MVC, lo que permite una separación clara de las responsabilidades dentro de la aplicación. El modelo maneja la lógica de datos y la interacción con la base de datos SQL, asegurando que la información se almacene y gestione de manera segura. La vista está compuesta por HTML, CSS y Bootstrap, proporcionando una interfaz de usuario moderna, responsive y fácil de usar. El controlador actúa como intermediario, gestionando la interacción entre el usuario y el sistema, y coordinando las acciones necesarias para realizar las votaciones de manera eficiente.

## Características Principales

### Interfaz de Usuario Amigable:

El sistema de votación está diseñado para ser intuitivo, permitiendo a los usuarios navegar y votar con facilidad. La interfaz, creada con Bootstrap, asegura que la aplicación sea accesible desde diferentes dispositivos y tamaños de pantalla.

### Gestión Eficiente de Votaciones:

El sistema permite a los usuarios registrarse, iniciar sesión y emitir su voto de manera segura. Cada votante puede ver a los candidatos disponibles y seleccionar su opción preferida. Una vez realizado el voto, el sistema se encarga de almacenarlo en la base de datos de manera segura y sin redundancias.

### Seguridad de Datos:

La implementación de SQL en el backend asegura que los votos se registren y almacenen de forma segura, evitando cualquier tipo de manipulación o pérdida de datos. Además, se incluyen medidas para garantizar que cada usuario solo pueda votar una vez, protegiendo la integridad del proceso electoral.

### Análisis y Resultados en Tiempo Real:

Una vez finalizado el proceso de votación, el sistema puede generar reportes en tiempo real mostrando los resultados de las votaciones, permitiendo a los administradores y a los usuarios ver de manera inmediata el desenlace de la elección.

## Beneficios del Sistema

Este sistema de votación no solo facilita la administración de procesos electorales, sino que también mejora la experiencia del votante al ofrecer una plataforma moderna, segura y accesible. La implementación de MVC con C# asegura que el sistema sea robusto y escalable, mientras que el uso de Bootstrap garantiza una interfaz atractiva y fácil de usar.

En resumen, este proyecto ofrece una solución completa para gestionar votaciones, integrando tecnología de punta para asegurar un proceso eficiente y amigable para todos los usuarios involucrados.

# Diagrama de flujo

## Esrtuctura del diagrama de Flujo

 Inicio**:**

* **Nodo de Inicio:** Representa el punto de partida del sistema.

 Configuración **de la Aplicación:**

* **Bloques de Proceso:**
  + **ConfigurarServicios:** Define y configura los servicios necesarios.
  + **ConfigurarMiddleware:** Configura las rutas y autorizaciones.

 Controladores**:**

* **Subdivisión por Controlador:**
  + **AdminControllers:**
    - **CreateCandidato, EditCandidato, CreateUsuario, EditUsuario, ResultadosFinal:** Procesos para gestionar candidatos y usuarios.
  + **AuthControllers:**
    - **Login:** Verifica credenciales.
    - **Register:** Registra nuevos usuarios.
  + **HomeController:**
    - **MostrarPaginaPrincipal:** Muestra la página de inicio.
    - **MostrarPoliticaPrivacidad:** Muestra la política de privacidad.
  + **VotanteControllers:**
    - **VerCandidatos:** Muestra los candidatos.
    - **EmitirVoto:** Registra el voto si el usuario no ha votado previamente.

 Base **de Datos y Modelos:**

* **ApplicationDbContext:** Maneja las conexiones y operaciones con la base de datos.
* **Modelos:**
  + **Candidato, Usuario, Voto, ResultadoVoto, ErrorViewModel, LoginViewModel:** Representan las entidades del sistema.

 Vistas**:**

* **AdminViews:**
  + **CreateCandidatoView, EditCandidatoView, CreateUsuarioView, EditUsuarioView, IndexAdminView, ManageUsersView, ResultadosFinalView:** Formularios y páginas para gestionar el sistema.
* **AuthViews:**
  + **LoginView, RegisterView:** Formularios de autenticación.
* **HomeViews:**
  + **IndexHomeView, PrivacyView:** Página principal y política de privacidad.
* **SharedViews:**
  + **LayoutView, ValidationScriptsPartialView, AccessDeniedView, ErrorView:** Plantillas compartidas y vistas de error.
* **VotanteViews:**
  + **ViewImportsView, ViewStartView:** Configuración de las vistas de votación.

 Fin**:**

* **Nodo de Fin:** Representa el punto donde el flujo del proceso termina.

|  |
| --- |
|  |

Se divide en dos el diagrama de flujo ya que lucidchart que es la herramienta utilizada para crear el diagrama, tiene una versión limitada de figuras, por lo cual se divide en dos, sin embargo se adjuntan ambos links de la misma, para corroboración del diagrama:   
  
diagrama1:<https://lucid.app/lucidchart/a78b46bb-b575-4eb8-88ee-e76949c06e45/edit?viewport_loc=-417%2C-401%2C2823%2C1414%2C0_0&invitationId=inv_bedaf371-3f30-400c-9a7a-94d897a6c0fd>

Diagrama2: <https://lucid.app/lucidchart/e2bc6cea-24df-40ae-9c2f-14091e729573/edit?view_items=_-22JFDz-K.y%2C_-22la897hnM%2C_-22OB4a64GR%2C_-22xlyvvRFT%2C_-22iSLbZ0Q-%2C_-22dmGSbqZF%2C_-22K41BpG7o%2C_-22a1ZUr8Fo%2C_-22C46_~Lxg%2C_-224i2ma5kT%2C_-22hBHw5A~V%2C_-22~Jc0ERlS%2C_-22zRk~xN7O%2C_-22Iv2l9O3G%2C_-22xiEZ5p3L%2C_-229dV5_FXt%2C_-222CuwxMPA%2C_-22DTOQhzU1%2C_-22JsPVjm._%2C_-22MEbOdKwh%2C_-22~v9Y8R21%2C_-22tONrexW5%2C_-229mr5dR.b%2C_-22rshfRfCg%2C_-228TsrNpbn%2C_-22zBRtajL2%2C_-22vh-.387c%2C_-22vEhS.pqn%2C_-22lkk8iefp%2C_-22UV~78KaJ%2C_-22hsIZUxy~%2C_-22Cyk90_hO%2C_-226I1dXH4w%2C_-22xIoAj~h3%2C_-2290.xblUj%2C_-22bKh1pEvY&invitationId=inv_4a89dba8-2d41-4bc7-b88d-146344591293>

|  |
| --- |
|  |

# Explicación sobre la base de Datos

## Analisis y Justificación

Para este proyecto, elegir una base de datos SQL en lugar de una NoSQL es la opción más adecuada por varias razones clave que se alinean con los requisitos y objetivos del proyecto.

En primer lugar, las bases de datos SQL proporcionan una estructura de datos consistente y bien definida. Dado que las bases de datos relacionales utilizan un esquema rígido, se garantiza que los datos se almacenen de manera organizada y coherente. Esto es esencial para proyectos donde las relaciones entre diferentes conjuntos de datos son críticas y deben mantenerse de manera consistente. Por ejemplo, en aplicaciones que manejan datos financieros, transacciones o cualquier información que requiera integridad y precisión, la capacidad de SQL para mantener la integridad referencial es fundamental.

En segundo lugar, las bases de datos SQL son altamente eficientes en la ejecución de consultas complejas. El lenguaje SQL está diseñado para realizar operaciones sofisticadas y joins entre múltiples tablas de manera efectiva. Esto es particularmente importante si tu proyecto necesita recuperar datos específicos de conjuntos de datos interrelacionados o ejecutar análisis complejos. La capacidad de realizar estas consultas de manera eficiente sin comprometer el rendimiento es una ventaja significativa de las bases de datos SQL.

Además, las bases de datos SQL soportan transacciones ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad). Estas propiedades son cruciales para aplicaciones que deben garantizar que todas las operaciones dentro de una transacción se completen correctamente y que los datos permanezcan consistentes en todo momento. Esto es especialmente importante en escenarios donde la integridad de los datos es primordial, como en aplicaciones bancarias, sistemas de reservas o cualquier otra aplicación donde las transacciones seguras y confiables son esenciales.

El mantenimiento y soporte son otras ventajas importantes de las bases de datos SQL. Estas bases de datos han existido durante décadas y tienen una amplia comunidad de usuarios, lo que significa que hay abundante documentación, herramientas de administración y soporte disponible. Esto facilita el mantenimiento continuo del sistema y la resolución de problemas que puedan surgir. Además, dado que el lenguaje SQL es un estándar ampliamente aceptado en la industria, es más fácil encontrar personal cualificado y herramientas que puedan integrarse sin problemas con otros sistemas y tecnologías.

En resumen, para un proyecto que requiere una estructura de datos bien definida, integridad referencial, soporte para transacciones ACID y la capacidad de ejecutar consultas complejas de manera eficiente, una base de datos SQL es la mejor opción. Proporciona la robustez, fiabilidad y soporte necesarios para garantizar el éxito del proyecto, asegurando que los datos se gestionen de manera segura y eficiente.

# Desarrollo de la Pagina Web

## Estructura del código

El proyecto ***ProyectoVotacion*** está diseñado utilizando el patrón de arquitectura MVC (Modelo-Vista-Controlador), lo que permite una separación clara entre la lógica de negocio, la interfaz de usuario y el control de la aplicación. A continuación, te presento una explicación detallada de la estructura del código y las carpetas que componen este proyecto.

### Carpeta Controllers

En esta carpeta se encuentran los controladores, los cuales son responsables de manejar las solicitudes de los usuarios, interactuar con los modelos, y devolver las vistas adecuadas. Los controladores disponibles en el proyecto son:

* **AdminControllers:** Controla las operaciones administrativas, como la creación y edición de candidatos y usuarios, la gestión de usuarios y la visualización de los resultados finales.
* **AuthControllers:** Maneja la autenticación de los usuarios, permitiendo el acceso al sistema a través de funciones como login y registro.
* **HomeController:** Controlador encargado de las páginas principales del sistema, como la página de inicio (index) y la política de privacidad.
* **VotanteControllers:** Controla las acciones relacionadas con los votantes, como la visualización de candidatos y la emisión de votos.

### Carpeta Data

Esta carpeta es crucial para la interacción con la base de datos y la gestión de la persistencia de los datos.

* **ApplicationDbContext:** Es el contexto de base de datos que gestiona la conexión con la base de datos y la interacción con los modelos. Define las tablas y sus relaciones en la base de datos, y es el punto central para las operaciones CRUD (Crear, Leer, Actualizar, Eliminar).

### Carpeta Migrations

Esta carpeta contiene las migraciones que son archivos generados por Entity Framework para reflejar los cambios en los modelos de datos y aplicarlos a la base de datos. Aquí se almacenan los scripts necesarios para mantener la sincronización entre la base de datos y el modelo de datos a medida que se realizan cambios en la estructura de datos del proyecto.

### Carpeta Models

En esta carpeta se encuentran los modelos que representan las entidades de datos del sistema, así como las clases que ayudan a gestionar los datos y la lógica de negocio.

* **Candidato.cs:** Representa a un candidato en el sistema, incluyendo propiedades como nombre, partido, etc.
* **ErrorViewModel.cs:** Modelo utilizado para gestionar y mostrar los errores en la aplicación.
* **LoginViewModel.cs:** Modelo utilizado en la vista de login para gestionar los datos de inicio de sesión del usuario.
* **ResultadoVoto.cs:** Modelo que representa los resultados de las votaciones.
* **Usuario.cs:** Representa a los usuarios del sistema, incluyendo propiedades como nombre, roles, etc.
* **Voto.cs:** Modelo que representa un voto emitido por un usuario, incluyendo información sobre el candidato seleccionado.

### Carpeta Views

En esta carpeta se encuentran las vistas, que son las plantillas de la interfaz de usuario que se renderizan y muestran al usuario final. Están organizadas en subcarpetas según los controladores a los que pertenecen:

* **Carpeta admin:** Contiene vistas relacionadas con las funciones administrativas, como la creación y edición de candidatos y usuarios, la gestión de usuarios, y la visualización de resultados.
  + **createcandidato:** Vista para crear un nuevo candidato.
  + **createusuario:** Vista para crear un nuevo usuario.
  + **editcandidato:** Vista para editar la información de un candidato.
  + **editusuario:** Vista para editar la información de un usuario.
  + **index:** Vista principal del área de administración.
  + **manageusers:** Vista para la gestión de usuarios.
  + **resultadosfinal:** Vista para mostrar los resultados finales de la votación.
* **Carpeta auth:** Contiene vistas relacionadas con la autenticación de usuarios.
  + **login:** Vista de inicio de sesión.
  + **register:** Vista de registro de nuevos usuarios.
* **Carpeta home:** Contiene vistas relacionadas con las páginas principales del sistema.
  + **index:** Vista de la página principal del sitio.
  + **privacy:** Vista de la política de privacidad.
* **Carpeta Shared:** Contiene vistas compartidas que son utilizadas en diferentes partes del sistema.
  + **layout:** Vista que define el layout general del sitio, incluyendo cabeceras, pie de página, y otras secciones comunes.
  + **validationscriptspartial:** Vista parcial que incluye scripts para validaciones en las vistas.
  + **accessdenied:** Vista que se muestra cuando un usuario intenta acceder a una página sin los permisos necesarios.
  + **error:** Vista que se muestra cuando ocurre un error en el sistema.
* **Carpeta Votante:** Contiene vistas relacionadas con las operaciones que pueden realizar los votantes.
  + **viewimports:** Archivo de importación de vistas que define las dependencias necesarias para las vistas de votantes.
  + **viewstart:** Archivo que define el layout o la plantilla que se utiliza en las vistas de votantes.

### Archivos Restantes o fuera de las carpetas

Además de las carpetas mencionadas, el proyecto incluye dos archivos importantes en el directorio raíz:

* **Program.cs:** Archivo principal que contiene el punto de entrada de la aplicación. Aquí se configura y arranca la aplicación, definiendo servicios, middleware, y la configuración inicial.
* **appsettings.json:** Archivo de configuración que contiene configuraciones globales del proyecto, como cadenas de conexión a la base de datos, configuraciones de autenticación, y otros ajustes clave para la aplicación.

En conjunto, esta estructura de carpetas y archivos facilita la organización del proyecto, permitiendo un desarrollo modular y escalable, al mismo tiempo que proporciona una experiencia de usuario fluida y eficiente.

# Resultados del Proyecto

El proyecto ***ProyectoVotacion*** ha sido diseñado para proporcionar un sistema de votación robusto y accesible, utilizando el patrón de arquitectura MVC en C#, junto con tecnologías como HTML, CSS, Bootstrap, y SQL para ofrecer una interfaz amigable y funcional. Los resultados esperados incluyen una plataforma segura y eficiente que permita a los usuarios registrar sus votos de manera transparente, gestionar candidatos y usuarios, y visualizar los resultados en tiempo real. El uso de modelos y controladores bien estructurados asegura que la lógica de la aplicación esté separada de la presentación, facilitando el mantenimiento y escalabilidad del sistema. En conclusión, el proyecto no solo cumple con los requisitos funcionales de un sistema de votación, sino que también ofrece una base sólida para futuras expansiones o adaptaciones según las necesidades de la comunidad usuaria.