Objetivos Técnicos:

1. **Seguridad Robusta:** Implementar medidas de seguridad avanzadas para proteger la integridad de los datos y garantizar la autenticidad de cada voto. La encriptación y autenticación serán fundamentales para prevenir posibles vulnerabilidades.
2. **Usabilidad Intuitiva:** Desarrollar una interfaz de usuario amigable e intuitiva que permita a votantes de todas las edades y habilidades participar sin dificultades. La accesibilidad será una prioridad, asegurando que el software sea inclusivo para todos.
3. **Escalabilidad:** Diseñar el software para adaptarse a diferentes escalas de elecciones, desde votaciones locales hasta procesos nacionales. La arquitectura del sistema deberá ser flexible y capaz de manejar cargas variables de manera eficiente.
4. **Transparencia y Auditoría:** Integrar herramientas que faciliten la auditoría y el seguimiento del proceso electoral. Cada etapa del proceso de votación será transparente y susceptible a revisión, fomentando la confianza en los resultados.
5. **Adaptabilidad Tecnológica:** Desarrollar el software con la capacidad de integrarse con tecnologías emergentes y futuras. Esto asegurará que el sistema de votación se mantenga actualizado y sea resistente a la obsolescencia.

Al abordar estos objetivos técnicos, aspiramos no solo a modernizar el proceso electoral, sino a sentar las bases para un sistema de votación confiable, inclusivo y transparente que refleje los principios fundamentales de la democracia.

Link del proyecto: alexandevcwa/VOTAGT: Sistema de votaciones digitales (github.com)

Diagramas:

Casos de Uso Link: VOTAGT/diagrams/Diagramas de Casos de Uso VOTAGT.svg at main · alexandevcwa/VOTAGT (github.com)

Diagrama de Clases: VOTAGT/diagrams/Diagramas de Clases.svg at main · alexandevcwa/VOTAGT (github.com)

Tecnologías utilizadas

1. **Java:** Como lenguaje de programación, Java es la columna vertebral de muchas aplicaciones empresariales y de software. Es conocido por su portabilidad, orientación a objetos y robustez.
2. **IntelliJ IDEA:** Es un entorno de desarrollo integrado (IDE) que ofrece soporte para varios lenguajes, pero es especialmente conocido por su excelente soporte para Java. Algunas características notables incluyen la autocompletación inteligente, refactorización de código, depuración avanzada y herramientas de análisis estático.

3. **Java Virtual Machine (JVM):** Es esencial para ejecutar aplicaciones Java. JVM permite la ejecución de código Java en cualquier sistema que tenga la JVM instalada, proporcionando así portabilidad de código.
4. **Maven o Gradle:** Estas son herramientas de gestión de proyectos que ayudan a manejar las dependencias y construir proyectos de manera eficiente. Maven y Gradle son muy utilizados en proyectos Java para gestionar las bibliotecas y automatizar el proceso de compilación.
5. **JUnit/TestNG:** Para realizar pruebas unitarias en Java, se utilizan bibliotecas como JUnit o TestNG. Estas permiten a los desarrolladores escribir pruebas automáticas para garantizar la calidad del código.
6. **Git:** Para el control de versiones, Git es ampliamente utilizado en proyectos Java. Permite realizar un seguimiento de los cambios en el código y colaborar de manera efectiva en equipos de desarrollo.

1. Instala Java:

Asegúrate de tener Java instalado en tu sistema. Puedes descargarlo desde java.com.

2. Instala Maven:

Descarga e instala Maven desde el sitio oficial de Apache Maven.

3. Configura las variables de entorno:

Añade las siguientes variables de entorno a tu sistema:

- **JAVA_HOME**: La ruta al directorio de instalación de Java.
- **MAVEN_HOME**: La ruta al directorio de instalación de Maven.
- Añade **%JAVA_HOME%\bin** y **%MAVEN_HOME%\bin** al **PATH** para acceder a los ejecutables de Java y Maven desde cualquier ubicación en la línea de comandos.

4. Verifica las instalaciones:

Abre una nueva ventana de terminal y ejecuta los siguientes comandos para verificar las instalaciones:

java -version

mvn -v

5. Crea un proyecto Maven:

Utiliza el siguiente comando para crear un nuevo proyecto Maven (reemplaza **com.example** y **mi-proyecto** con tu propio identificador de paquete y nombre de proyecto):

mvn archetype:generate -DgroupId=com.example -DartifactId=mi-proyecto -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

6. Navega al directorio del proyecto:

cd mi-proyecto

7. Compila el proyecto:

Ejecuta el siguiente comando para compilar el proyecto:

mvn compile

8. Ejecuta las pruebas:

mvn test

9. Empaqueta el proyecto:

mvn package

Esto generará un archivo JAR en el directorio **target**.

10. Ejecuta el proyecto:

java -jar target/mi-proyecto-1.0-SNAPSHOT.jar

Asegúrate de reemplazar **mi-proyecto-1.0-SNAPSHOT.jar** con el nombre real de tu archivo JAR.

Sección de solución de porblemas

1. Error al descargar dependencias:

- **Problema:** Maven no puede descargar las dependencias del repositorio remoto.
- **Solución:** Verifica tu conexión a internet. Ejecuta **mvn clean install -U** para forzar la actualización de las dependencias.

2. Problemas de compatibilidad de versiones:

- **Problema:** Incompatibilidad entre las versiones de las dependencias.
- **Solución:** Utiliza la propiedad **<maven.compiler.source>** y **<maven.compiler.target>** en tu archivo **pom.xml** para especificar la versión de Java que estás utilizando.

<properties>

  <maven.compiler.source>1.8</maven.compiler.source>

  <maven.compiler.target>1.8</maven.compiler.target>

</properties>

3. Error de clase no encontrada (ClassNotFoundException):

- **Problema:** Maven compila correctamente, pero al ejecutar, no encuentra una clase.

- **Solución:** Asegúrate de que todas las dependencias necesarias estén en el classpath. Puedes usar el plugin de Maven para ejecutar tu aplicación:

xml

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.1</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.codehaus.mojo</groupId>
      <artifactId>exec-maven-plugin</artifactId>
      <version>3.0.0</version>
      <executions>
        <execution>
          <goals>
            <goal>java</goal>
          </goals>
        </execution>
      </executions>
      <configuration>
        <mainClass>com.example.MainClass</mainClass>
      </configuration>
    </plugin>
```

```
    </plugins>
</build>
```

4. Problema con el repositorio Maven central:

- **Problema:** Maven no puede descargar dependencias desde el repositorio central.
- **Solución:** Verifica tu conexión a internet y asegúrate de que Maven esté configurado para usar el repositorio central en tu **settings.xml**:

```
<mirrors>
    <mirror>
        <id>central</id>
        <url>https://repo.maven.apache.org/maven2</url>
        <mirrorOf>central</mirrorOf>
    </mirror>
</mirrors>
```

Información de contactarlo

EMAIL: alexandermachic.0@gmail.com Tel: 1234-5678