A password strength generator

Arquitectura e integración de Sistemas Software
2023/2024

Javier Castilla Rodríguez (javcasrod1@alum.us.es)

Francisco Gago Vázquez (fragagvaz@alum.us.es)

Carlos Galea Magro (cargalmag@alum.us.es)

Miguel Soriano Rodríguez (migsorrod1@alum.us.es)

Tutor: Irene Bedilia Estrada Torres

Número de Grupo: IS2 L4-7

Enlace a GitHub: https://github.com/nulab/zxcvbn4j/tree/main

HISTORIAL DE VERSIONES

Fecha	Versión	Descripción	Participantes
29 Febrero 2024	1.0	Versión Inicial	Javier Castilla Carlos Galea Francisco Gago Miguel Soriano
7 Marzo 2024	1.1	Contribuciones, documentación y sugerencias de mejora.	Javier Castilla Carlos Galea Francisco Gago Miguel Soriano
10 Marzo 2024	2.0	Versión final	Javier Castilla Carlos Galea Francisco Gago Miguel Soriano

ÍNDICE

INTRODUCCIÓN

zxcvbn4j es la versión de zxcvbn para Java, un calculador de seguridad de contraseñas, originalmente escrito en JavaScript e inspirado por herramientas de descifrado de contraseñas. Utiliza el ajuste de patrones y la estimación conservadora para evaluar la fuerza de las contraseñas. [1]

También reconoce palabras populares en inglés de Wikipedia, programas de televisión estadounidenses y películas. Además, puede detectar otros patrones de contraseñas comunes como fechas, caracteres repetidos (aaa), secuencias (abcd), patrones de teclado (qwertyuiop) y L33t Speak (o Leet Speak, es una forma de escritura alternativa que utiliza caracteres alfanuméricos para reemplazar letras)

Figura 1. *Atributos de popularidad*

Podemos saber que el proyecto se creó el 24 de diciembre de 2015 ya que ahí se realizó el primer commit y que sigue activo siendo el último commit el 7 de noviembre de 2023 que fue a su vez el último pull-request. El proyecto cuenta con 35 releases, el último fue la versión 1.8.2 el 21 de agosto de 2023. La última issue fue cerrada el 12 de enero de 2024, Existen issues más recientes sin cerrar, aunque todas son en concepto de mejoras para el proyecto.

VISIÓN GENERAL

A continuación, mostramos un diagrama que representa el funcionamiento general de nuestra librería:

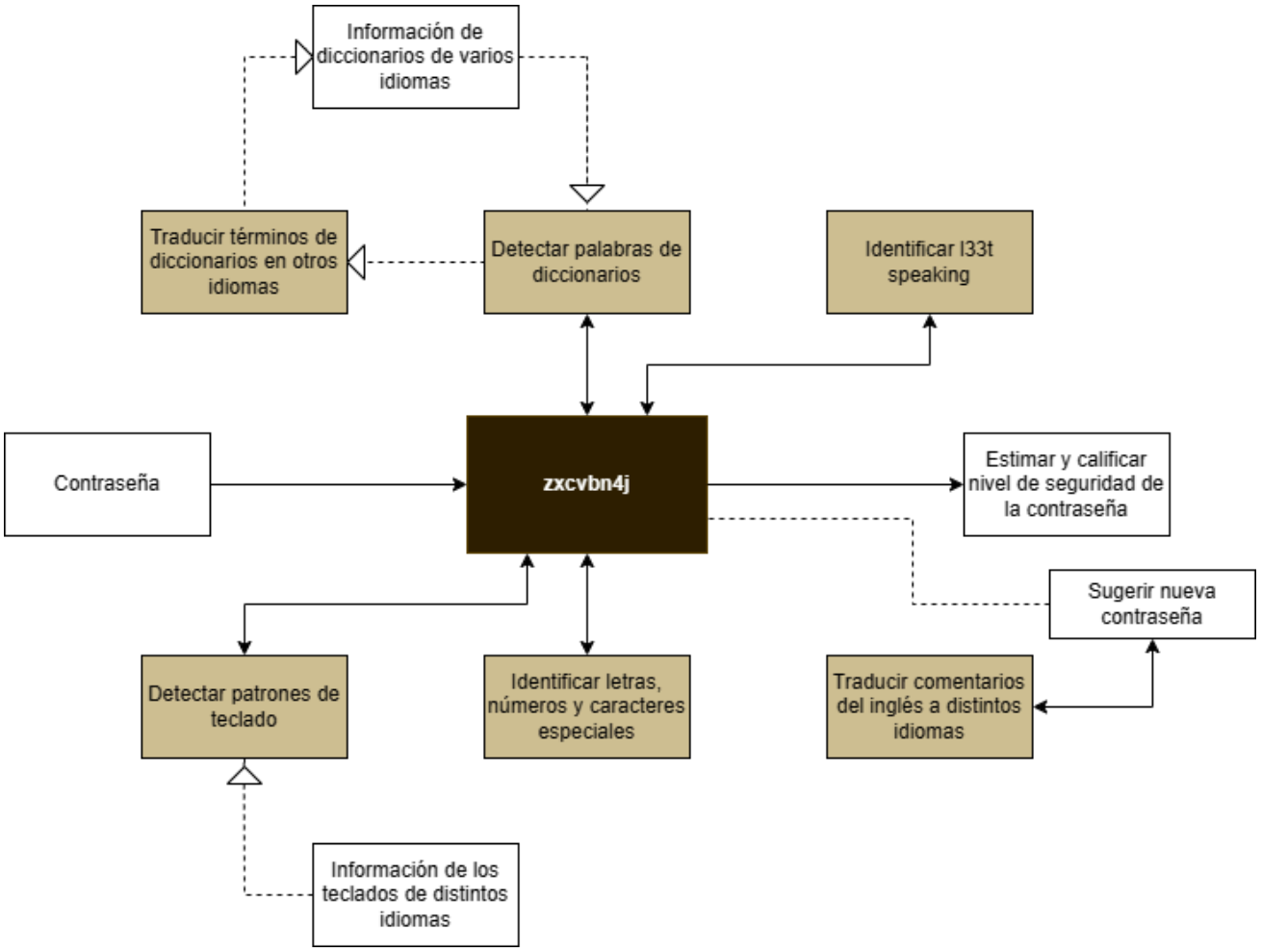

Figura 2. *Visión general del sistema*

Su funcionamiento se basa en calificar la seguridad de las contraseñas detectando patrones de teclado, fechas, caracteres repetidos, palabras de diccionarios e identificando l33t (leet speaking, basado en la sustitución de letras por otros caracteres).
Para ello, se apoya en la información de diccionarios (nombres, apellidos, palabras de la wikipedia,series de tv, ...) y de teclados de distintos idiomas (qwerty, dvorak, jis, ...), y tras comprobar si la contraseña coincide con alguno de los patrones anteriormente descritos, hace una estimación de su fuerza, calificándola y aportando mensajes de feedback en distintos idiomas.

PARTICIPANTES

Para la clasificación de participantes hemos usado la clasificación de participantes de Wood. [2]
En la siguiente tabla encontramos a algunos de los contribuidores:

Nombre	Nombre usuario github
Yuichi Watanabe	@vvatanabe
Yasuyuki Baba	@yasuyuki-baba
Sebastian Stenzel	@overheadhunter
Maxim Mazine	@mazine
Kana	@kxn4t
Keisuke Kato	@kasecato
+20	+20

Los participantes que más han contribuido son los siguientes:

- **vvatanabe**: es la persona que más commits ha hecho, tanto de arreglo de errores, como commits sobre el readme y refactorización, con un total de 172 commits. También fue quién realizó el primer commit, por lo que podemos considerarlo como el desarrollador principal del sistema.
Además, se ha encargado de aceptar los merge pull request de las personas interesadas en contribuir al proyecto y de hacer commits sobre las versiones del programa. Por lo tanto, podemos suponer que ejerce tanto el rol de supervisor del proyecto como el de encargado de la documentación y comunicador.

- **yasuyuki-baba**: es el segundo contribuidor con más commits realizados, con un total de 22 commits. Se ha encargado de mejorar el rendimiento del programa y corregir errores en el código. Tiene asignado el rol de desarrollador del proyecto.

- **overheadhunter**: además de realizar varios cambios en el código, se encargó principalmente de controlar las versiones de Java y Gradle necesarias para el correcto funcionamiento del programa. Se le puede considerar tanto desarrollador del proyecto como personal de soporte.

- **mazine**: podemos ver que fue quién se encargó de atender los primeros merge pull request y arreglar los primeros errores. Además, añadió una localización personalizable para los comentarios, así como varios tests. Tiene asignado el rol de desarrollador del proyecto.

VISTAS

Vista de contexto

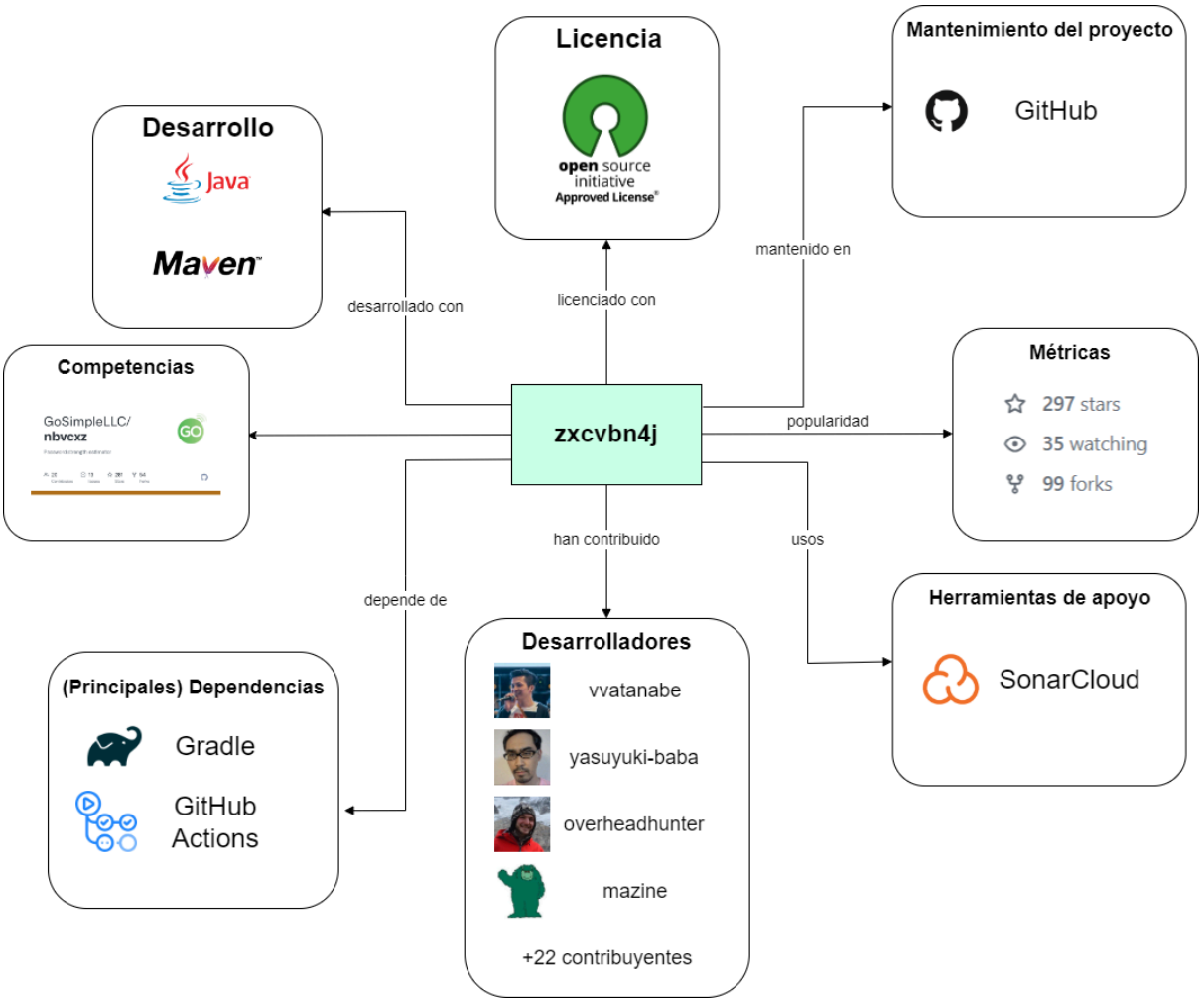

Figura 3. *Vista de contexto del sistema [3]*

La vista de desarrollo describe las relaciones, dependencias e interacciones entre el sistema y su entorno (personas, sistemas y entidades externas con las que interactúa).

El proyecto ha sido realizado y desarrollado con Java al 100% con apoyo de Maven Central que sirve de apoyo para la gestión del proyecto, facilita el mantenimiento y gestión de dependencias de proyecto, además descarga las bibliotecas necesarias para el proyecto, este programa también está escrito en Java según lo proporcionado en GitHub. Donde está alojado el mantenimiento del proyecto. [4]

Las principales dependencias del proyecto son Gradle y GitHub Actions. Gradle es una herramienta de automatización de construcción y gestión de dependencias, es muy usado para proyectos Java. En este proyecto aporta definición y gestión de las dependencias del proyecto, compilar el código fuente, ejecutar pruebas, y crear distribuciones del software,

entre otras tareas de construcción. En general sirve para resolver las dependencias de tu proyecto y asegurar que todas las bibliotecas y recursos necesarios estén disponibles para tu aplicación durante la construcción y ejecución. [5]

Por otro lado GitHub Actions es un servicio de automatización integrado en GitHub, en el proyecto ayuda con la automatización de tareas, algo de gran importancia en el proyecto ya que nos aporta también garantización de calidad del código

La herramienta de apoyo principal es Soñar Cloud, que ayuda a la recopilación de datos varios del proyecto y ayuda al seguimiento y calidad del código, facilitando la localización de puntos fuertes y débiles en el código del proyecto.

Con respecto a sus métricas podemos ver que tiene 297 estrellas de popularidad, 35 personas siguen el proyecto, por tanto les interesa y tiene 99 forks. Hay que tener en cuenta que el proyecto sigue en desarrollo y sus métricas de popularidad irán variando según su avance.

El proyecto usa una MIT License, una licencia de código abierto que permite el libre uso , modificación y distribución de software y otorga permisos muy amplios a los usuarios. Muy utilizada y conocida en proyectos software.

Como competencia directa tiene una, llamada nbvcxz, la cual se inspiró en el proyecto zxcvbn y trató de mejorarlo. [6]

Han trabajado 26 desarrolladores en total. vvatanabe realizó 172 commits, seguido de yasuyuki-baba con 22 commits y overheadhunter con 21. El resto de los desarrolladores también han contribuido pero su cantidad de commits es mucho menor.

Vista de escenarios

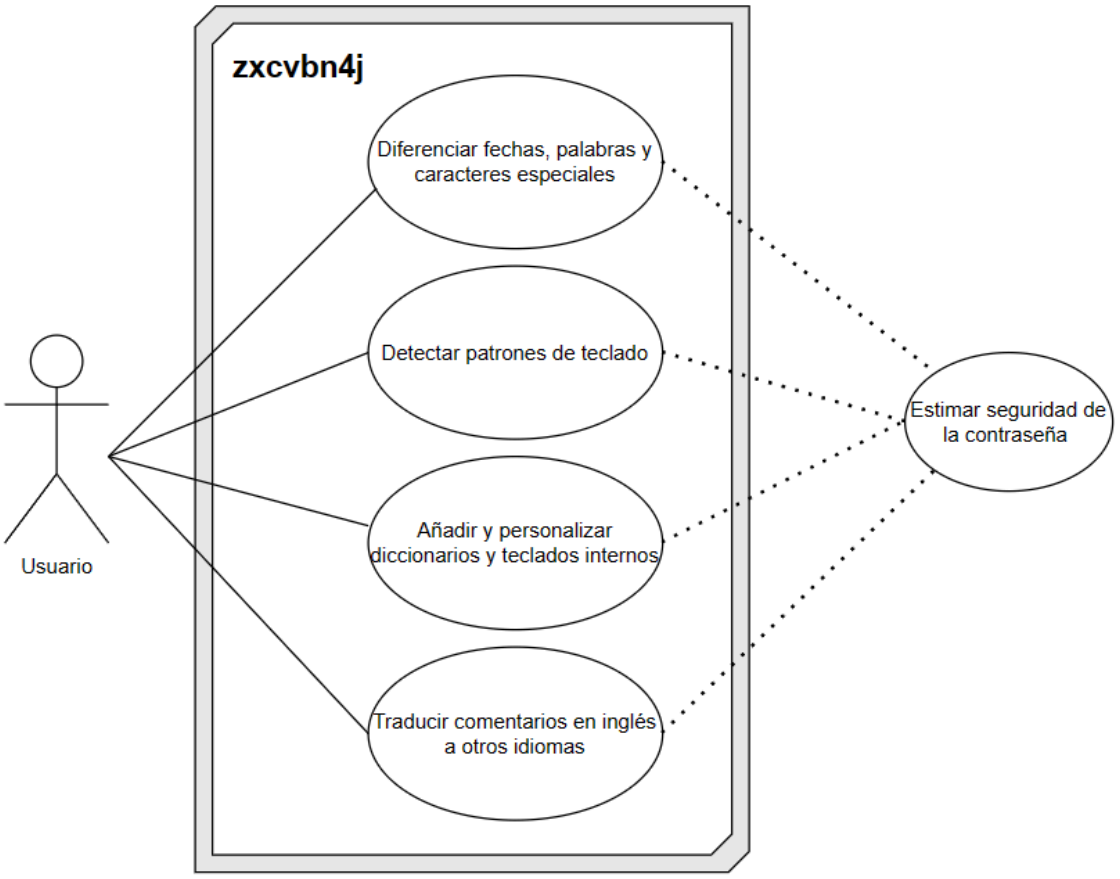

Figura 4. *Diagrama UML de casos de uso.[3]*

Los escenarios de uso describen para que se usará el sistema, en este se han identificado los siguientes casos de uso:

• Diferenciar fechas, palabras y caracteres especiales: El programa tiene la capacidad de detectar contraseñas específicas y conocidas, por tanto es útil a la hora de detectar fechas, palabras registradas del diccionario y caracteres especiales.

• Detectar patrones de teclado: Al igual que detecta caracteres singulares, también tiene la capacidad de detectar los patrones de teclados introducidos por el usuario. Algunos patrones son “xcvb” o “1234”. A partir de estos puede sugerir mejoras.

• Añadir y personalizar diccionarios y teclados internos: El programa puede agregar sus propios diccionarios en su sistema para quedar registradas esas palabras y así tenerlas por si alguna vez se introducen.

• Traducir comentarios en inglés a otros idiomas: Al introducir una palabra en inglés, el programa la guarda y la puede traducir a otros idiomas.

Vista funcional

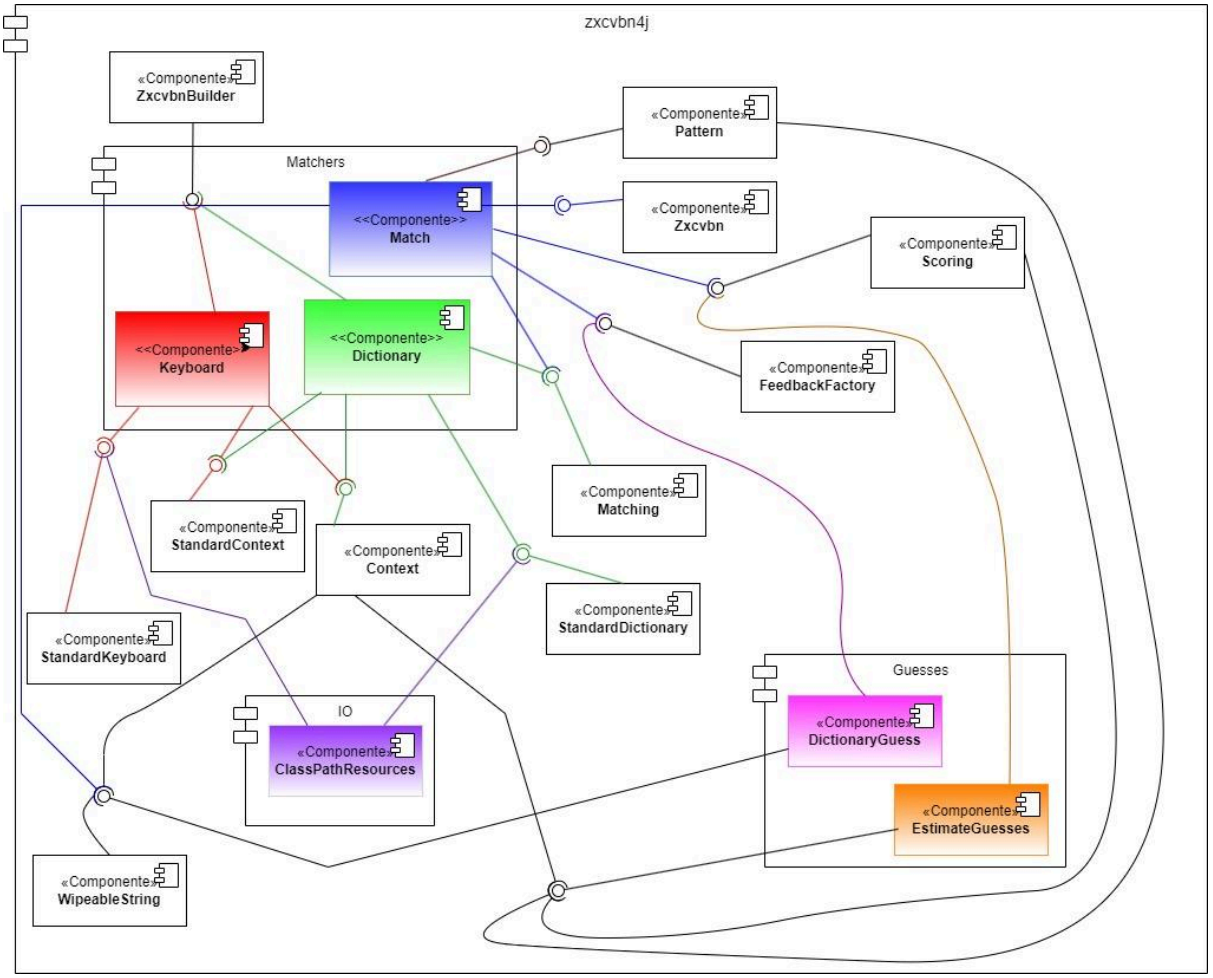

Figura 5. *Diagrama UML de componentes [3]*

La vista funcional describe los componentes, responsabilidades, interfaces e interacciones principales del sistema.
En el caso de nuestro proyecto, podemos ver que la mayoría de componentes interaccionan con el componente Match, por lo que el sistema depende altamente de él.

Desde el punto de vista funcional, podemos identificar los siguientes componentes principales:

Matchers: Este componente es el encargado de detectar diferentes patrones en nuestra contraseña. Incluye numerosos subcomponentes que tienen diferentes funciones:

- Construir gráficos que diferencian caracteres cercanos en los teclados.
- Detectar fechas en diferentes formatos.
- Identificar secuencias al buscar repeticiones, también sobre textos con otros caracteres como los alfabetos griego y cirílico.
- Obtener las coordenadas adyacentes de una tecla en un teclado.

Guesses: sus clases implementan una estrategia que estiman el número de intentos que requeriría adivinar una contraseña. Cada una de ellas se centra en estimar un patrón diferente.

IO: el código de este componente es un framework de Spring. En primer lugar, devuelve el InputStream a utilizar; normalmente, el ClassLoader del contexto del hilo, si está disponible. A continuación, se utiliza el ClassLoader que cargó la clase ResourceLoader. Por último, si ni siquiera el ClassLoader del sistema puede acceder al recurso, se devuelve null. [7]

Vista de despliegue

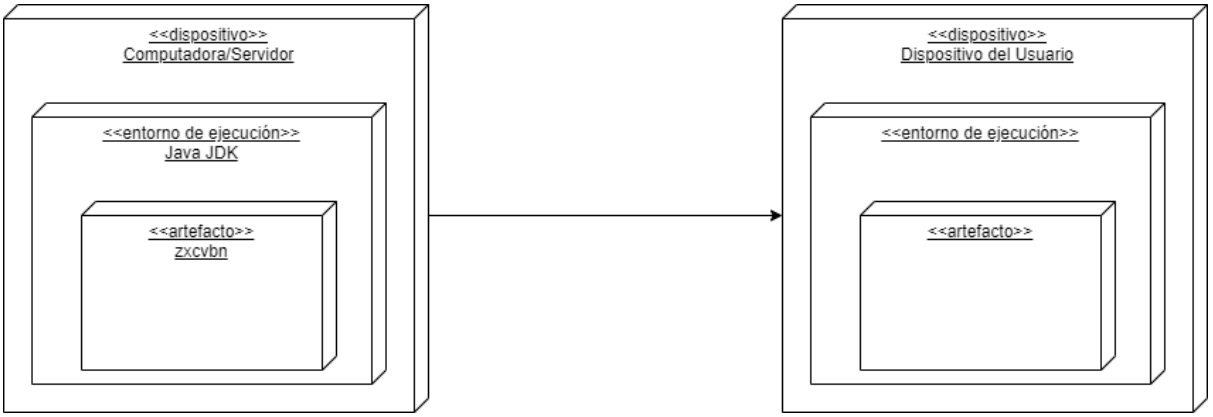

Figura 6. *Diagrama UML de despliegue [3]*

La vista de despliegue describe el entorno hardware y software en el que se desplegará el sistema, incluyendo sus dependencias en el entorno de ejecución.

La implementación solo requiere una instalación de Java JDK que es el entorno de ejecución (software) para poder instalar la librería zxcvbn. Mediante conexiones HTTP se consigue ejecutar la librería en el dispositivo del usuario.

Vista de desarrollo

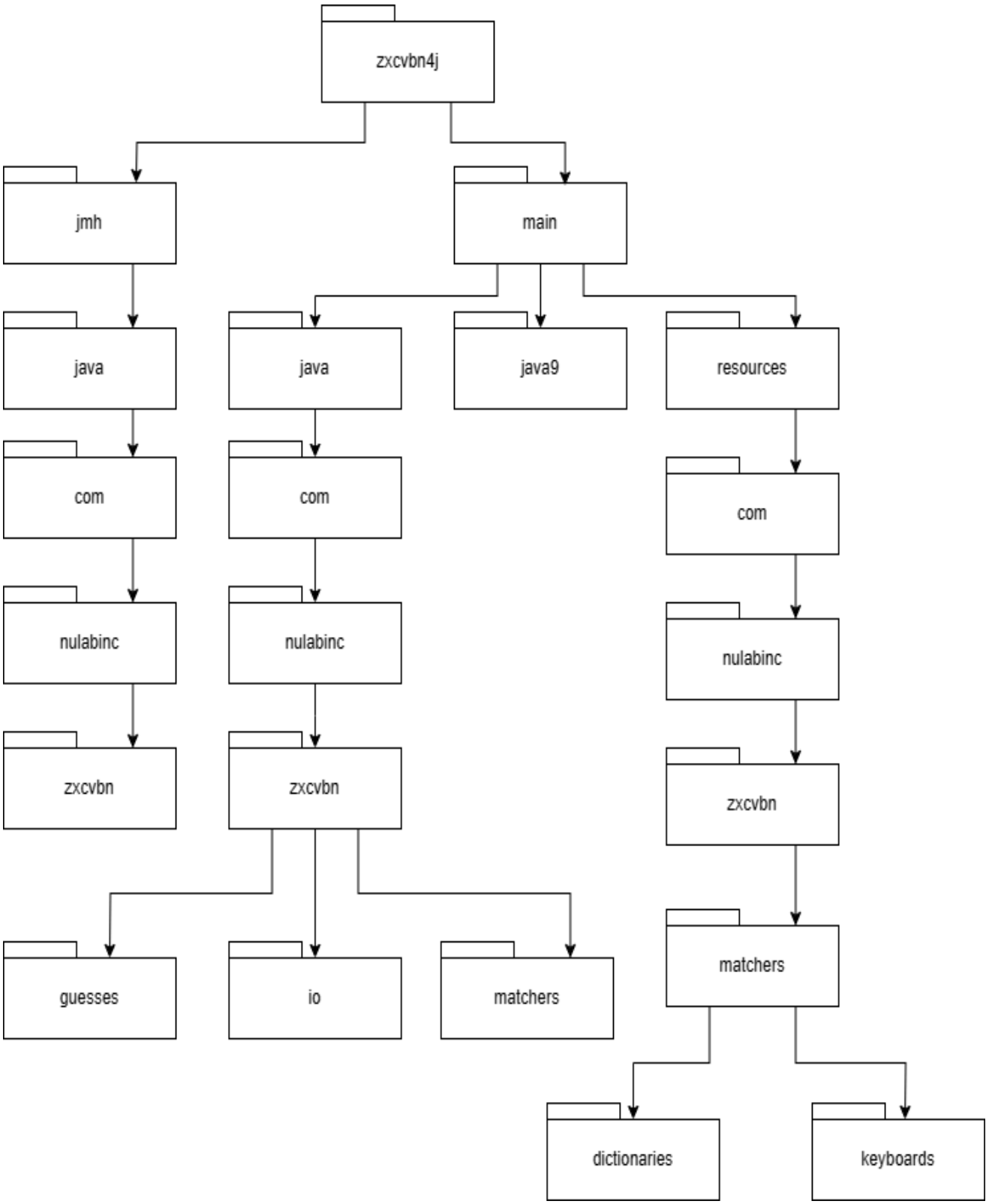

Figura 7. *Diagrama UML de paquetes [3]*

La vista de desarrollo describe la arquitectura que da soporte al proceso de desarrollo software.
Los paquetes que conforman esta vista están definidos por funcionalidad.

Específicamente, el proyecto define dependencias con las siguientes bibliotecas principales:

- **source:** Directorio donde se encuentra la mayoría de archivos.
  - **jmh/java/com/nulabinc/zxcvbn:** Contiene el archivo RandomPasswordMeasureBenchmark.java, el cual mide el rendimiento con el que se evalúa la seguridad de las contraseñas.
  - **main:** Contiene los tres paquetes principales del sistema.
    - **java/com/nulabinc/zxcvbn:** Contiene la mayoría de archivos .java utilizados para definir las funcionalidades del sistema, también contiene tres paquetes con más código.
      - **guesses:** Contiene las clases que estiman el número de intentos para adivinar las contraseñas.
      - **io:** Contiene dos archivos en los cuales se describen las excepciones que se lanzarán en el caso de que suceda algo no previsto.
      - **matchers:** Contiene los archivos cuyo código describe cómo enlazar los componentes entre ellos.
    - **java9:** Contiene el module-info.
    - **resources/com/nulabinc/zxcvbn:** Contiene archivos .properties de cada uno de los idiomas disponibles, los cuales definen los mensajes que se muestran al usuario tras introducir una contraseña.
      - **matchers:**
        - **dictionaries:** archivos .txt con las palabras que detecta el sistema.
        - **keyboard:** archivos .txt con los formatos de teclado que detecta el sistema.

PUNTOS DE VARIABILIDAD Y EXTENSIÓN

Puntos de variabilidad

Los puntos de variabilidad son partes del sistema en las que pueden variar y donde podemos optar por diferentes alternativas llamadas variantes. Hemos encontrado los siguientes en el código del programa:

Motor de Javascript: El desarrollador puede cambiar el motor de JavaScript para facilitar la integración de scripts en Java durante las pruebas.

```
33
34      dependencies {
35          testImplementation group: 'junit', name: 'junit', version: '4.13.2'
36          testImplementation group: 'org.graalvm.js', name: 'js', version: '22.0.0.2'
37          testImplementation group: 'org.graalvm.js', name: 'js-scriptengine', version: '22.0.0.2'
38          jmh group: 'org.openjdk.jmh', name: 'jmh-core', version: '1.34'
39          jmh group: 'org.openjdk.jmh', name: 'jmh-generator-annprocess', version: '1.34'
40          jmh group: 'org.ow2.asm', name: 'asm', version: '9.3'
41      }
```

Figura 8. *Código para cambiar el motor de JavaScript*

Lo podemos localizar en el archivo build.gradle en la carpeta raíz del proyecto.

Teclados y diccionarios: los usuarios tienen permiso para utilizar sus propios teclados con cualquier distribución y libertad para personalizar el diccionario (el que usa el programa para valorar la contraseña con palabras del diccionario.

```
1   package com.nulabinc.zxcvbn;
2
3   import com.nulabinc.zxcvbn.matchers.Dictionary;
4   import com.nulabinc.zxcvbn.matchers.Keyboard;
5   import java.util.Collections;
6   import java.util.Map;
7
8   public class Context {
9
10    private final Map<String, Dictionary> dictionaryMap;
11
12    private final Map<String, Keyboard> keyboardMap;
13
14    public Context(
15        final Map<String, Dictionary> dictionaryMap, final Map<String, Keyboard> keyboardMap) {
16      this.dictionaryMap = dictionaryMap;
17      this.keyboardMap = keyboardMap;
18    }
19
20    public Map<String, Dictionary> getDictionaryMap() {
21      return Collections.unmodifiableMap(this.dictionaryMap);
22    }
23
24    public Map<String, Keyboard> getKeyboardMap() {
25      return Collections.unmodifiableMap(this.keyboardMap);
26    }
27  }
```

Figura 9. *Código para utilizar teclados con cualquier distribución*

Este punto de variabilidad se encuentra en zxcvbn4j/src/jmh/java/com/nulabinc/zxcvbn, donde se contiene el archivo Context.java.

Puntos de extensión

Los puntos de extensión permiten la modificación o ampliación del sistema sin necesidad de modificar su código fuente principal. Permiten agregar nuevas funcionalidades o comportamientos sin afectar la lógica existente del programa. Algunos de los que hemos encontrado son:

Maven-publish: Proporciona la capacidad de publicar artefactos en repositorios Maven.

Signing: Habilita la firma de artefactos.obertura de código con JaCoCo.

Checkstyle: Configura la verificación del estilo de código.

Kt3k.coveralls: Habilita la integración con Coveralls para informes de cobertura.

Champeau.jmh: Proporciona integración con JMH (Java Microbenchmarking Harness).

Github.spotbugs: Configura la detección de errores potenciales con SpotBugs.

Sherter.google-java-format: Configura el formato de código según las convenciones de estilo de Google.

```
1   plugins {
2       id 'java-library'
3       id 'maven-publish'
4       id 'signing'
5       id 'jacoco'
6       id 'checkstyle'
7       id 'com.github.kt3k.coveralls' version '2.12.2'
8       id "me.champeau.jmh" version "0.7.1"
9       id "com.github.spotbugs" version "5.1.3"
10      id 'com.github.sherter.google-java-format' version '0.9'
11  }
```

Figura 10. *Plug-ins añadidos a los puntos de extensión*

Lo podemos localizar en el archivo build.gradle en la carpeta raíz de proyecto

ANÁLISIS DE ATRIBUTOS DE CALIDAD

Los atributos de calidad que hemos analizado para este proyecto son: [8]

Fiabilidad

Se refiere al correcto funcionamiento del sistema cuando se está ejecutando.
El proyecto tiene una muy buena fiabilidad con una calificación de “A” por SonarCloud con 0 bugs identificados.

Mantenibilidad

Es la facilidad con la que se puede cambiar el código para introducir mejoras o corregir errores.
El proyecto tiene una buena mantenibilidad, consiguiendo calificación según SonarCloud de “A”. A pesar de ello se detectan 26 Code Smells en el código, que equivalen a una deuda técnica de cuatro horas y cuarto.

Seguridad

Se define como la capacidad del sistema para evitar que se pueda acceder sin permiso a los datos que almacena.
La seguridad del sistema también cumple con una calificación de “A” en Sonarcloud debido a que la sencillez y uso del programa hace que no se requiera almacenar información privada y se puede considerar un problema menor del que ocuparse.

Rendimiento

Se refiere a los recursos informáticos utilizados por el sistema como tiempo de ejecución, consumo de memoria o uso de energía.
Tiene un rendimiento bastante alto debido a la sencillez del programa.

Usabilidad

Facilidad que se les presenta a los usuarios para utilizar el sistema.
La usabilidad del proyecto es baja, con respecto a que presenta muchos problemas de instalación y no facilita el uso a los distintos usuarios, la documentación del repositorio es mediocre, además tiene 7% de porcentaje de líneas de código comentadas.

SUGERENCIAS DE MEJORA

- **Upgrade del Gradle Wrapper:** el wrapper de Gradle en el script de construcción del proyecto está actualmente configurado a la versión 8.3. Recomendaría actualizarlo a la última versión, la 8.6, para aprovechar las nuevas características, mejoras de rendimiento y correcciones de errores proporcionadas por las versiones más recientes de Gradle.

- **APIs externas:** habilitar el Matcher utilizando APIs externas como una característica opcional.

- Sanitized inputs: las contraseñas introducidas como sanitized inputs hacen que algunas funciones de cálculo de seguridad se comporten de forma extraña.

- **Entropías precisas:** la puntuación obtenida de las calculadoras de seguridad son un poco abstractas, y estaría bien que mostrasen el valor preciso de entropía de la contraseña para proporcionar una mejor información al usuario.

- Los nombres de las funciones y las variables deben ser **descriptivas.**

CONTRIBUCIONES AL PROYECTO

Nuestra aportación al proyecto ha sido añadir el portugués a los distintos idiomas disponibles para dar feedback. Simplemente habría que añadir el archivo **messages_pr.properties** (imagen a continuación) en la ruta "src/main/resources/com/nulabinc/zxcvbn". No es necesario modificar código ya que podemos ver que el archivo Feedback.java, encargado de identificar los distintos idiomas disponibles para el feedback, detecta todos los archivos de la ruta "com/nulabinc/zxcvbn/messages".

```
+ # Feedback
+ ## Extra
+ feedback.extra.suggestions.addAnotherWord=Adicione outra palavra. Palavras incomuns s\u00E3o melhores.
+
+ ## Default
+ feedback.default.suggestions.useFewWords=Use v\u00E1rias palavras, evite frases comuns.
+ feedback.default.suggestions.noNeedSymbols=Voc\u00EA n\u00E3o precisa de s\u00EDmbolos, d\u00EDgitos ou letras mai\u00FAsculas.
+
+ ## Dictionary
+ feedback.dictionary.warning.passwords.top10=Essa senha est\u00E1 entre as 10 mais comuns.
+ feedback.dictionary.warning.passwords.top100=Essa senha est\u00E1 entre as 100 mais comuns.
+ feedback.dictionary.warning.passwords.veryCommon=Esta \u00E9 uma senha muito comum.
+ feedback.dictionary.warning.passwords.similar=Isso \u00E9 semelhante a uma senha comumente usada.
+ feedback.dictionary.warning.englishWikipedia.itself=Uma palavra por si s\u00F3 \u00E9 f\u00E1cil de adivinhar.
+ feedback.dictionary.warning.etc.namesThemselves=Somente o nome e o sobrenome s\u00E3o f\u00E1ceis de adivinhar.
+ feedback.dictionary.warning.etc.namesCommon=Nomes e sobrenomes comuns s\u00E3o f\u00E1ceis de adivinhar.
+ feedback.dictionary.suggestions.capitalization=A capitaliza\u00E7\u00E3o n\u00E3o ajuda muito.
+ feedback.dictionary.suggestions.allUppercase=Escrever tudo em letras mai\u00FAsculas \u00E9 quase t\u00E3o f\u00E1cil de adivinhar quanto escrever tudo em letras min\u00FAsculas.
+ feedback.dictionary.suggestions.reversed=As palavras ao contr\u00E1rio n\u00E3o s\u00E3o muito mais dif\u00EDceis de adivinhar.
+ feedback.dictionary.suggestions.l33t=Substitui\u00E7\u00F5es previs\u00EDveis como '@' em vez de 'a' n\u00E3o ajudam muito.
+
+ ## Spatial
+ feedback.spatial.warning.straightRowsOfKeys=As linhas de teclas retas s\u00E3o f\u00E1ceis de adivinhar.
+ feedback.spatial.warning.shortKeyboardPatterns=Padr\u00F5es curtos de teclado s\u00E3o f\u00E1ceis de adivinhar.
+ feedback.spatial.suggestions.UseLongerKeyboardPattern=Use um padr\u00E3o de teclado mais longo com mais voltas.
+
+ ## Repeat
+ feedback.repeat.warning.likeAAA=Repeti\u00E7\u00F5es como "aaa" s\u00E3o f\u00E1ceis de adivinhar.
+ feedback.repeat.warning.likeABCABCABC=Repeti\u00E7\u00F5es como "abcabcabc" s\u00E3o apenas um pouco mais dif\u00EDceis de adivinhar do que "abc".
+ feedback.repeat.suggestions.avoidRepeatedWords=Evite palavras e caracteres repetidos.
+
+ ## Sequence
+ feedback.sequence.warning.likeABCor6543=Sequ\u00EAncias como "abc" ou "6543" s\u00E3o f\u00E1ceis de adivinhar.
+ feedback.sequence.suggestions.avoidSequences=Evite sequ\u00EAncias.
+
+ ## Regex
+ feedback.regex.warning.recentYears=Os \u00FAltimos anos s\u00E3o f\u00E1ceis de adivinhar.
+ feedback.regex.suggestions.avoidRecentYears=Evite os \u00FAltimos anos e os anos que est\u00E3o relacionados a voc\u00EA.
+
+ ## Date
+ feedback.date.warning.dates=As datas costumam ser f\u00E1ceis de adivinhar.
+ feedback.date.suggestions.avoidDates=Evite datas e anos relacionados a voc\u00EA.
```

Figura 11. *Feedback de zxcvbn4j en portugués*

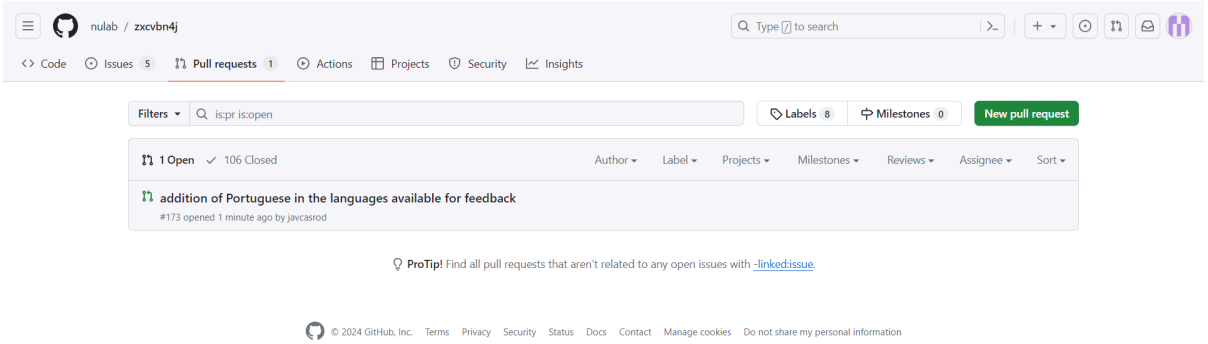

Figura 12. *Pull request #173*

CONCLUSIONES

En este proyecto, hemos documentado el diseño arquitectónico de Zxcvbn4j, una versión de Zxcvbn para Java y sirve para calcular la seguridad de contraseñas, desarrollada principalmente por Yuichi Watanabe.

La librería tiene un diseño de alto acoplamiento (explicar por qué tiene acoplamiento, es porque sus módulos dependen mucho unos de otros como hemos visto en el diagrama de componentes) y es extensible, demostrando un buen rendimiento en términos de métricas de calidad estándar. Sin embargo, hay espacio para mejorar aspectos como la documentación y el diseño.

La principal limitación probablemente es la dependencia del componente Match, lo que podría causar problemas en el futuro para desarrollar, probar y mantener los diferentes módulos.

REFERENCIAS

[1] Dropbox-zxcvbn (proyecto de Github)
https://github.com/dropbox/zxcvbn

[2] Software Systems Architecture. (s. f.). 2005 - 2011 Nick Rozanski and Eoin Woods

https://www.viewpoints-and-perspectives.info/home/stakeholders/

[3] Herramienta usada para crear los diagramas UML de las vistas:
draw.io
https://app.diagrams.net/

[4] Sonatype-Maven Central Repository. xzcvbn.

https://central.sonatype.com/artifact/com.nulab-inc/zxcvbn/1.8.2

[5] Gradle Build Tool
https://gradle.org/

[6] nbvcxz
https://github.com/GoSimpleLLC/nbvcxz

[7] Spring Framework
https://github.com/spring-projects/spring-framework/tree/dfb7ca733ad309b35040e0027fb7a2f10f3a196a

[8] SonarCloud
https://sonarcloud.io/summary/overall?id=AISSProjects24_zxcvbn4j