

Project Report

Juego de preguntas LaEspiral
v0.1, January 21, 2023

by

Amaia Echeandia Paredes

Contents

1	Introducción	1
1.1	Introducción	1
1.2	Descripción del proyecto	1
2	Arquitectura	4
2.1	Introduction	4
2.2	Visión general de la arquitectura	4
2.3	Evaluación de riesgos	5
2.4	Nueva arquitectura propuesta	5
3	Diseño	7
3.1	Intro	7
3.2	Atacantes	7
3.3	Criterio de diseño hasta la fecha	7
3.4	Evaluación de riesgos	8
3.5	Nuevo diseño propuesto	9
3.6	Cuestionario de diseño	9
4	Implementation	10
4.1	Introducción	10
4.2	Lenguaje de programación y herramientas	10
4.3	Descripción del código	10
4.4	Fallos en el código	11
4.4.1	Análisis estático	12
4.5	Valoración de riesgos	13
4.6	Implementación propuesta	14
5	Operaciones	15
5.1	Intro	15
5.2	Deficiencias.	15
5.3	Mejoras	15
6	Automatización	17
6.1	Introducción	17
6.2	Herramientas	17
6.3	Pruebas funcionales	17
7	Conclusión	20
	Appendices	21
A	Cuestionario	22
B	Diagrama de clases	31

1

Introducción

Anyone:
Repositorio Github: <https://github.com/AmaiaEtxe/DPS.git>

1.1. Introducción

Este documento recoge el análisis de seguridad de una aplicación desarrollada por dos estudiantes de la Universidad ORT Uruguay.

El planteamiento del informe sigue los cinco principios básicos de la ciberseguridad que recoge el libro *Secure coding: principles and practices* de los autores *Kenneth R. Van Wyk, Mark Graff*. En este libro explican y aplican las siguientes fases:

1. Architecture
2. Design
3. Implementation
4. Operations
5. Automation and test

Estos principios tienen como objetivo asegurar la calidad del software y practicar un desarrollo seguro. Y aunque el desarrollo seguro de aplicaciones debería estar ya asociado a la calidad del software, todavía se puede encontrar buen software que no cumple unos principios básicos de ciberseguridad.

1.2. Descripción del proyecto

En esta sección, se hará una descripción del funcionamiento del proyecto. Como ya hemos comentado, este proyecto pertenece a dos estudiantes de la Universidad ORT Uruguay donde hicieron el desarrollo cursando la asignatura de Ingeniería de Software. Esta aplicación consiste en un juego llamado *LaEspiral-2.0* de tipo quiz con diferentes estilos de preguntas y lo podemos encontrar en el repositorio de GitHub mencionado al principio de la introducción.

Como objetivo, esta aplicación se basa en una serie de quizzes para el uso en un contexto educativo, aunque se podría utilizar para otros aspectos.

A grandes rasgos, el juego consiste en cargar una serie de preguntas de diferentes formatos e ir respondiendo a las preguntas en un tiempo limitado de tiempo. La aplicación se ejecutaría de manera local y con un solo usuario.

Para hacer una carga de preguntas, el usuario deberá proporcionar un archivo .txt escrito en formato GIFT. Este formato lo que permite es escribir preguntas de tipo Verdadero/Falso, múltiple opción, respuestas breves, palabra faltante y preguntas numéricas. En nuestro caso, esta aplicación estará preparada para soportar los tres primeros formatos de pregunta. A continuación se detalla el estilo que debería seguirse para las tres preguntas REFERENCIA:

```
// falso/verdadero
::Q1:: 1+1=2 {T}

// opción múltiple con retroalimentación especificada para respuestas correctas e incorrectas
::Q2:: ¿Qué color hay entre el naranja y el verde en el espectro?
{ =Amarillo# correcto; bien! ~rojo # incorrecto, es amarillo ~azul # incorrecto, es amarillo }

// respuesta corta
::Q3:: ¿Cómo estás? {}
```

Además, antes de cada pregunta, se podrá especificar el tiempo que deseas aplicar a cada pregunta (ej: //15). Aunque si no es el caso, existe ya un tiempo predeterminado, que son 30s.

Es importante recalcar que el número máximo de preguntas será de 39.

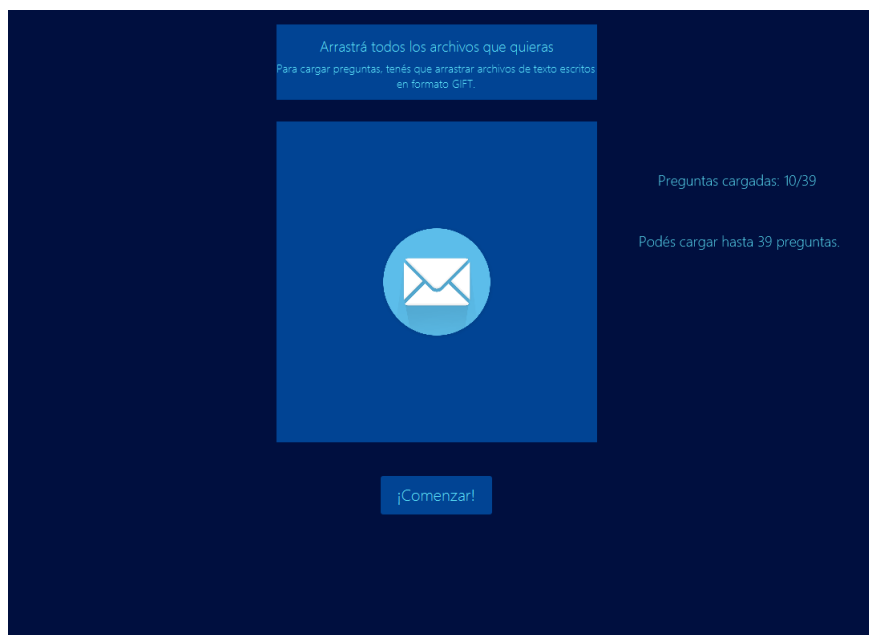


Figure 1.1: Interfaz gráfica de usuario con las preguntas cargadas

Una vez cargado este archivo o archivos, se mostrará un tablero con forma de espiral la cual contará con unas casillas, representando en gris claro cada una de las preguntas, y en gris oscuro las casillas que no contienen ninguna pregunta cargada (si aplica). Por lo que la cantidad de casilleros habilitados será la misma que preguntas cargadas.

El juego comenzará cuando el usuario se meta en una de las casillas y continuará contestando una a una cada pregunta cargada. A medida que se vaya contestando, se irán visualizando las respuestas como correctas

o como fallidas y se mostrará un GIF se reproducirá un sonido dependiendo también de la respuesta dada.

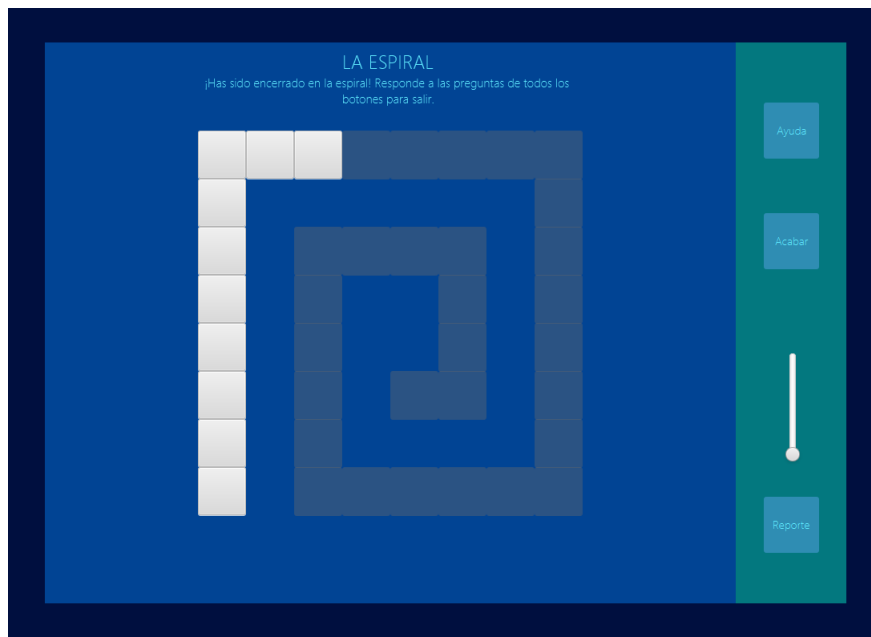


Figure 1.2: Interfaz gráfica de usuario con las preguntas cargadas



Figure 1.3: Ejemplo de pregunta corta

Al finalizar el quiz (incluso durante), habría posibilidad de generar un informe con todos los detalles de las respuestas. Este reporte sería en formato PDF y se guardaría en la carpeta del proyecto en `\Obligatorio\src\main\resources`.

Todas las interacciones con el sistema se realizan en una sola ventana, sin utilizar diálogos o ventanas emergentes.

2

Arquitectura

2.1. Introduction

En este primer capítulo se analizarán aspectos relacionados con la arquitectura del proyecto. Para evaluar si la arquitectura es robusta y segura primeramente haremos un estudio de los riesgos que actualmente tiene el planteamiento y después, se harán unas propuestas de arquitectura.

2.2. Visión general de la arquitectura

La definición de la arquitectura en el proyecto se puede entender como una arquitectura básica. Podemos ver en la imagen que la aplicación se desarrolla en un mismo equipo de manera local.



Figure 2.1: Arquitectura del sistema

Tenemos a un usuario que deslizará el archivo .txt de formato GIFT como entrada después de iniciar el juego y como salida tendría un fichero en formato .pdf en caso de que el jugador decidiera sacar un reporte de lo jugado. Por lo tanto, el perfil del usuario será de una persona con nociones muy básicas de cómo usar un ordenador. Lo único que tendrá que aprender es a escribir en formato GIFT para crear las preguntas y respuestas posibles de entrada a la aplicación.

A pesar de que los archivos deban estar en un mismo directorio, existe la excepción del fichero de entrada. Este fichero puede ubicarse en cualquier ubicación del sistema ya que la aplicación no interactúa con el, si no que coge los valores de entrada. También será necesario tener instalado Java Run Environment (JRE) para

poder ejecutar el archivo .jar directamente sin necesidad de IDEs.

Si el reporte es requerido por el usuario, este se guardará en la carpeta `\src\main\resources` con el nombre de `reporte.pdf`.

2.3. Evaluación de riesgos

Explicaremos las carencias que se han ido viendo en la arquitectura y se explicará por qué son puntos importantes:

- Escalabilidad. El objetivo es meramente educativo e interactivo, por lo que el uso de la aplicación dependerá de la institución o centro que utilice el juego. A día de hoy muchos de los recursos educativos apuestan por hacer uso de recursos en la web, por lo que se podría considerar la opción de implementar esta funcionalidad, lo que, a su vez, supondría un número mayor de usuarios utilizando el servicio de manera simultánea o incluso extendiendo el juego añadiendo participantes en un mismo quiz.
- Seguridad en la red. No hay reglas declaradas que controlen la red y restrinjan posibles amenazas ya que se ejecuta de manera local.
- Copia de seguridad. El programa no hace ninguna copia de seguridad por lo que si se diera el caso de algún error y o el programa se cerrara por algún motivo, este no guardará las modificaciones hechas y se tendría que volver a comenzar el juego. Como ya hemos mencionado, sí que existe la posibilidad de recoger las respuestas en un reporte, pero esto solo soluciona el problema en caso de que hubieras hecho click en el botón justo antes del problema, y además, tendrías que rehacer las preguntas de igual manera.
- Control de acceso. No existe ninguna política de control de acceso para esta aplicación. Cualquier usuario tendrá acceso no solo al ejecutable si no al resto de archivos, como el fichero de entrada o el informe de respuestas.
- Archivos de log. No existe ningún archivo donde se recojan todas las jugadas a lo largo del tiempo.
- Control de archivos maliciosos. La aplicación recibe el fichero de entrada de manera que pueda desempeñar las funcionalidades. Al ser un archivo con contenido aleatorio que escribiría un usuario ajeno al desarrollo, el programa podría recibir cualquier formato de archivo como input. En cambio, el proyecto no contiene ningún tipo de control frente a posible malware.

2.4. Nueva arquitectura propuesta

Para cada una de las posibles vulnerabilidades encontradas haremos una propuesta de mejora. No podemos asegurarnos completamente de cubrir todos los fallos de seguridad ya que esto requeriría de más personas que revisaran el proyecto, pero sí que podremos acercarnos a una arquitectura aún más robusta.

- Escalabilidad. Hemos mencionado la implementación del juego en la web y con ello mayor número de usuarios. Para ello, se empezaría por establecer planes y estrategias con el objetivo de abordar y valorar mayor consumo de recursos tanto económicamente como tecnológicamente sin afectar a la seguridad.
- Seguridad en la red. Considerando una posible subida a internet, tendremos que proteger el sistema para que no se pueda acceder a puertos no relacionados con el servidor web desde el cliente. También sería buena opción hacer uso de Iptables para el supervisar el tráfico del servidor.
- Copia de seguridad. Para que se guarde la información después de un fallo en el software, es necesario el uso de un backup automático de los datos y guardarlos localmente, como el uso de la herramienta `rsync` o `FreeFileSync`. Dada la magnitud del proyecto, no veo la necesidad de almacenamiento externo para este tipo de aplicaciones ya que no tiene más objetivo que el de educar. En caso de que el fallo sea de hardware y el equipo entero esté dañado, las preguntas respondidas quedarían sin guardarse. Volviendo al tema de la aplicación web, esto podría cambiar si se sincronizara con copias de seguridad en la nube o en almacenamientos externos.

- Control de acceso. Las personas autorizadas a ejecutar el juego tendrían un usuario con su contraseña correspondiente siguiendo los requisitos mínimos de contraseña segura, esto implica: combinación de letras (mayúsculas/minúsculas), números y símbolos del estándar ASCII, y mínimo de 10 caracteres. Para guardar esta información sería necesario el uso de una base de datos de la cual se hablará más tarde.
- Archivos log. Debe de haber un archivo .log donde se registre la actividad del quiz y sus fallos.
- Control de archivos maliciosos. Conforme al nivel de aplicación, no vería necesario hacer una evaluación más allá del nombre del archivo, el tamaño y la extensión para ver que todo está correcto. Si el sistema fuera más complejo y tuviera la necesidad de conexión a la red, se podrían proponer soluciones como antivirus, o las comentadas en el punto de *Seguridad de red*.

3

Diseño

3.1. Intro

El segundo principio de la ciberseguridad se centra en el diseño del software. El planteamiento óptimo del diseño aprovecha al máximo las características del sistema y hará que se desarrolle de manera más ordenada y con pasos más firmes.

3.2. Atacantes

Es importante tener en cuenta el tipo de atacantes a la hora de hacer el diseño de la aplicación para poder anticiparse a los peligros. Estos atacantes se distinguen por los siguientes aspectos:

- El propio desarrollador
- Ataques directos o indirectos provenientes de algún malware
- Crimen organizado con alguna motivación
- Atacantes independientes sin motivo claro
- Atacante sin cualificación (Script kiddie)

Teniendo en cuenta el tipo de proyecto que estamos auditando, podemos decir que la persona o grupo podría atacar a modo de boikot contra la aplicación ya que la información que maneja no es de carácter sensible. Dentro de esta descripción podríamos considerar los atacantes sin ninguna motivación económica, desarrolladores o personal descontento o incluso propios usuarios de la aplicación como los Script Kiddies ya mencionados. También sería posible malware que de manera indirecta afectara el funcionamiento del programa.

3.3. Criterio de diseño hasta la fecha

En general y dado que el proyecto no es complejo, considero que el diseño del mismo fue cuidadosamente planificado. Para comenzar el proyecto se decidió hacer uso de la plataforma **GitHub** para el **control de versiones** con Git. Esta herramienta permitió lo siguiente:

- Crear una rama master donde se aloja el código fuente definitivo
- Hacer modificaciones en el código de manera colaborativa

- Poder guardar las diferentes versiones en caso de necesitar arreglar modificaciones o consultar versiones anteriores
- Añadir las dependencias y sus configuraciones
- Añadir elementos para la GUI
- Añadir pruebas
- Crear manual de usuario

Para **gestionar** mejor el proyecto, se hizo uso de la herramienta **Maven** la cual, a través de un archivo XML llamado POM que contiene la información general del proyecto, nos guía para poder estructurar todos los elementos del proyecto en nodos: configuraciones, dependencias...

Las etapas del ciclo de construcción de Maven garantizan solidez al proyecto y las más relevantes son las siguientes:

1. Validación de información necesaria
2. Compilación
3. Pruebas unitarias de código compilado
4. Empaquetar código (JAR en nuestro caso)
5. Verificación de integración
6. Instalación en el repo local para poder usarlo como dependencia en otros proyectos
7. Despliegue del paquete final para compartirlo con otros desarrolladores

3.4. Evaluación de riesgos

A pesar de haberse planeado el diseño de una manera cuidada, no cabe duda que siempre se pueden implementar mejoras en el futuro.

En este caso, podríamos listar unos cuantos riesgos o malas prácticas durante el desarrollo de la aplicación:

- **Requisitos.** No se describen requisitos previos al diseño. Los requisitos resultan indispensables en cualquier proyecto ya que permiten exponer los mínimos necesarios y también incluso limitar las funciones del proyecto por temas económicos, de logística u otros.
- **Planificación.** Tampoco se menciona ningún tipo de metodología seguida durante el trabajo. De esta manera, no podemos asegurarnos de que en cada fase del proyecto se hayan completado todos los requisitos o tareas necesarias y en el orden correcto.
- **Plan de recuperación.** No existe ningún plan de acción en caso de que la aplicación falle.
- **Respuesta ante incidentes.** Relacionado con el punto anterior, al tratarse de un proyecto tan pequeño, no existe un protocolo o plataforma para dar respuesta ante errores, incidentes o cuestiones que puedan surgir.
- **Documentación.** Existe un manual de usuario en PDF para la carga de preguntas, pero la explicación se encuentra un tanto escueta si el jugador no está muy familiarizado con las tecnologías. Explica correctamente los pasos para ejecutar y cargar el cuestionario pero, por ejemplo, no es muy intuitiva la sintaxis del fichero de preguntas para que puedas diseñarlo a tu manera y tampoco es clara la descripción para responder a las preguntas y finalizar el cuestionario.
- **Capacidad de archivos.** No se da ninguna información acerca del tamaño máximo de los archivos que se pueden subir y de los que el propio quiz puede llegar a generar. por lo que podría generar una sobrecarga en el sistema y colapsar la aplicación.

3.5. Nuevo diseño propuesto

El diseño propuesto en algunos casos no es corregible después de la finalización de la primera versión del juego, pero sí es aplicable para futuros cambios. Presentamos las soluciones a los riesgos expuestos en el anterior capítulo:

- **Requisitos.** Antes de proceder a la división de tareas, actividades, material... se tendrá que hacer una lista de requisitos cubriendo cada una de las partes que impliquen a la aplicación.
- **Planificación.** Es indispensable que todo desarrollo de SW lleve una planificación de tareas y gracias a los requisitos documentados (si es que lo están), podemos dividir las funcionalidades en equipos de trabajo. Esto nos facilita la optimización del tiempo puesto que podremos definir criticidad, tiempos, asignaciones de responsables y de fases, gestión de errores... Hay muchos SW de planificación de proyectos: JIRA, Redmine, Github...
- **Plan de recuperación.** Deberá de definirse un plan para recuperar la normalidad del juego en caso de falla.
- **Respuesta ante incidentes.** Deberá diseñarse un plan para abordar los posible problemas que surjan durante la ejecución, así como un boton o un link que te lleve a una plataforma para ponerte en contacto con el servicio de mantenimiento de la aplicación. También es posible crear una pagina de FAQ para preguntas técnicas de fácil solución.
- **Documentación.** Se debería de completar la guía de usuario con explicaciones claras de las siguientes secciones:
 - Archivo de entrada: tipo de preguntas que admite el programa, nomenclatura y sintaxis del formato GIFT y formato de archivo (.txt en nuestro caso)
 - Ejecución del juego: instalación de programas para la ejecución (en este caso sería un entorno de ejecución Java).
 - Carga de preguntas y presentación del tablero
 - Reglas y orden del juego: objetivos, descripción los botones disponibles...
 - Finalización del juego
- **Capacidad de archivos.** En el propio juego, ya se indica la cantidad de preguntas máximas del archivo que el jugador puede subir pero deberá también indicar la capacidad máxima de carga. Como el reporte generado por el propio juego está directamente relacionado con el fichero que el usuario sube a la app, el tamaño de archivo que se descarga no deberá diferir mucho al de carga.

3.6. Cuestionario de diseño

Para analizar el diseño con más detalle, disponemos de unos cuestionarios de evaluación de seguridad ya predefinidos con una serie de preguntas que nos ayudan a identificar las debilidades. Haremos uso del cuestionario *INFORMATION SECURITY REVIEW QUESTIONNAIRE* incluido en el Anexo A y sugerido en la asignatura.

Este cuestionario lo considero bastante completo. Está separado en hasta siete aspectos y, aunque estuviera diseñado específicamente para el departamento de soporte TI de City University of New York, se puede extrapolar a más sistemas. Además dispone de cuestionarios de respuesta abierta para que se puedan considerar todas las opciones.

4

Implementation

4.1. Introducción

Siguiendo la misma línea que en los dos primeros apartados, se hará un resumen del estado en el que se encuentra actualmente la herramienta y cuál fue su implementación en su momento. Posteriormente, se describirán los fallos encontrados, tanto funcionales como de código con un análisis estático, y se podrán concluir los riesgos y sus posibles soluciones.

4.2. Lenguaje de programación y herramientas

El lenguaje de programación utilizado en este proyecto decidió realizarse en *Java 8*. Es un lenguaje que te permite desarrollar código en cualquier sistema operativo lo cual da mucha versatilidad y es una de las características más importantes de Java. Es muy útil, además, para asegurar un código robusto.

El desarrollo de este código fue hecho de manera ordenada. Oracle recoge un documento donde describe un estándar de convenciones de Java, llamado **Java Code Conventions (JCC)**. Los puntos más importantes que se han considerado para la realización del proyecto fueron los siguientes:

- **Orden en la estructura de las clases.** Al ser un lenguaje orientado a objetos es de vital importancia el cuidado de la estructura de las clases. Esto implica añadir comentarios explicativos, declarar la clase y considerar variables, constructores y métodos.
- **Nomenclatura.** La denominación de las variables debe ser clara que exprese sin complejidad lo que está representando. Los nombres deben de ser comprensibles para cualquier desarrollador. Esto también aplica al nombre de las clases, métodos, etc.
- **Indentación.** Una buena indentación permite que quién lo lea, automáticamente ordene el código, entienda la estructura general e identificar bloques. Además, para el propio desarrollador también es de gran utilidad hacer uso de esta recomendación puesto que es más fácil identificar errores sintácticos.

Además de esto, el IDE utilizado fue **NetBeans** y junto con él, el plugin **CheckStyle** usado para asegurar mayor **calidad** del código.

4.3. Descripción del código

En esta sección, explicaremos la estructura de los diferentes archivos que componen este programa. También se podrá ver en el Anexo B un diagrama con todas las clases y sus interdependencias.

Nombre	Tipo	Nombre	Tipo	Nombre	Tipo
controladores	Carpeta de archivos	Pregunta.java	Archivo JAVA	VentanaPrincipalController.java	Archivo JAVA
dominio	Carpeta de archivos	PreguntaCortaRespuesta.java	Archivo JAVA		
FXMLController.java	Archivo JAVA	PreguntaMultipleOpcion.java	Archivo JAVA		
MainApp.java	Archivo JAVA	PreguntaVF.java	Archivo JAVA		
		Sistema.java	Archivo JAVA		

Figure 4.1: Archivos del código fuente

Dentro de la carpeta podremos encontrarnos con el archivo principal donde empezará la ejecución del programa. Este archivo se llama *MainApp.java* y el propósito es lanzar la aplicación JavaFX desde un archivo FXML. En esta misma ruta se encuentra el archivo *FXMLController.java* que inicializará la interfaz de usuario.

Dentro de la carpeta "controladores", veremos un solo archivo, *VentanaPrincipalController.java*, que corresponde con la ventana que se despliega al ejecutarse el juego. Este archivo contiene la clase *VentanaPrincipalController* y hace uso de botones, diferentes paneles, y donde incluso incorpora paquetes de audio.

Volviéndonos a la carpeta anterior y entrando a *dominio* nos encontramos con las siguientes carpetas:

Cada una de las carpetas corresponde con una clase:

- **Pregunta.java** define la clase llamada *Pregunta* y representa la estructura de las preguntas que tiene el quiz. Esta estructura contiene un string con la pregunta en cuestión, el tiempo en el que debe ser respondida, una tabla hash para la respuestas entre otros.
- **PreguntaCortaRespuesta.java** define la clase llamada *PreguntaCortaRespuesta* que representa la pregunta tipo corta y hereda las propiedades y métodos de la clase *Pregunta*.
- **PreguntaMultipleOpcion.java** define la clase llamada *PreguntaMultipleOpcion* que representa la pregunta con multiples opciones y espera del usuario una unica selección. Tiene una propiedad adicional para que se añada un comentario por cada respuesta. También hereda las propiedades y métodos de la clase *Pregunta*.
- **PreguntaVF.java** define la clase llamada *PreguntaVF* que, al igual que las clases anteriores, representa una pregunta de verdadero o falso y devuelve una representación de cadena de la pregunta, tiempo y respuesta correcta.
- **Sistema.java** define la clase llamada *Sistema* que tiene como función procesar el archivo y extraer las preguntas y respuestas con diferentes métodos.

4.4. Fallos en el código

La funcionalidad del juego tal y como se plantea al principio del proyecto no es del todo la correcta. Hay fallos en la implementación del código que hace que el sistema tenga los siguientes defectos:

- Cuando dos preguntas de respuesta corta se encuentran seguidas, el campo de la segunda respuesta contiene por defecto el texto de la anterior.
- Si se presiona el botón de "Ayuda" el timer sigue descontando tiempo.
- El deslizador de la derecha no tiene ninguna funcionalidad. la idea era utilizarlo para el volumen pero no fue implementado del todo.
- Si ejecutamos el archivo *LaEspiral-2.0.jar* no funciona el sonido. El sonido solo funciona desde el IDE.

4.4.1. Análisis estático

Para profundizar un poco más en errores o warnings del código, analizaremos el proyecto con la herramienta PMD. La herramienta nos permite encontrar problemas, ineficiencia en el código, generar informes del análisis, encontrar código duplicado, incluso te hace sugerencias de mejora.

Actualmente estamos utilizando el IDE Eclipse donde es posible instalar el plugin PMD y de esta manera, poder visualizar mejor los mensajes que en una terminal.

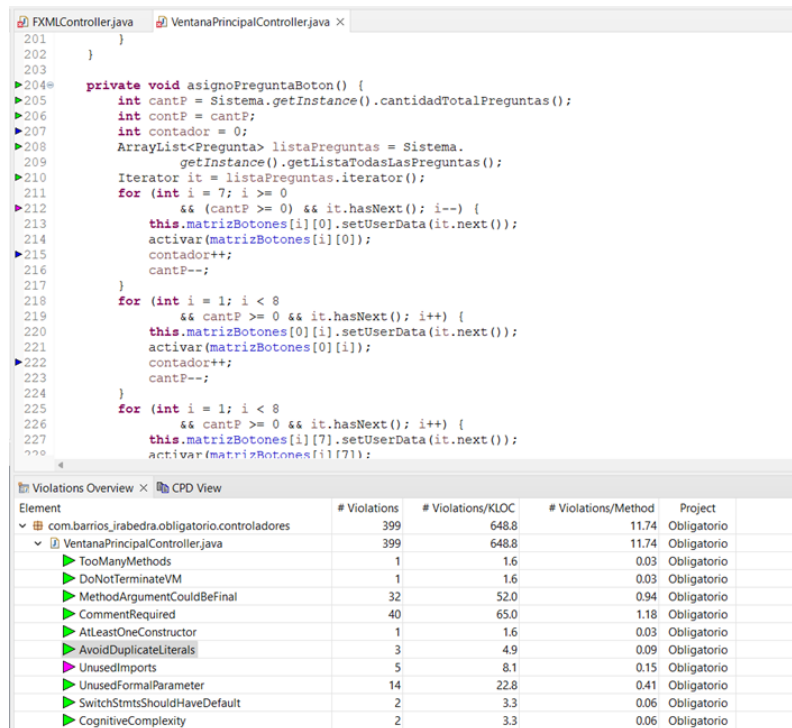


Figure 4.2: Archivos del código fuente

Violations Outline			
P	Line	Rule	Error Message
▶	83	CommentRequired	CommentRequired: Field comments are required
▶	105	LongVariable	LongVariable: Avoid excessively long variable names like txtPreguntasDropeadas
▶	113	CommentRequired	CommentRequired: Field comments are required
▶	603	EmptyCatchBlock	EmptyCatchBlock: Avoid empty catch blocks
▶	696	UnusedPrivateMethod	UnusedPrivateMethod: Avoid unused private methods such as 'onActionContinuarEnGifInCorrecto(ActionEvent)'.
▶	415	MethodArgumentCouldBeFinal	MethodArgumentCouldBeFinal: Parameter 'event' is not assigned and could be declared final
▶	591	AvoidCatchingGenericException	AvoidCatchingGenericException: Avoid catching generic exceptions such as NullPointerException, RuntimeException,...
▶	524	LongVariable	LongVariable: Avoid excessively long variable names like respuestaSeleccionada
▶	706	UnusedPrivateMethod	UnusedPrivateMethod: Avoid unused private methods such as 'onActionBotonAyuda(ActionEvent)'.
▶	357	AccessorMethodGeneration	AccessorMethodGeneration: Avoid autogenerated methods to access private fields and methods of inner / outer clas...
▶	146	LocalVariableCouldBeFinal	LocalVariableCouldBeFinal: Local variable 'files' could be declared final
▶	589	LocalVariableCouldBeFinal	LocalVariableCouldBeFinal: Local variable 'audioOk' could be declared final
▶	208	LocalVariableCouldBeFinal	LocalVariableCouldBeFinal: Local variable 'listaPreguntas' could be declared final

Figure 4.3: Archivos del código fuente

Una vez ejecutada la herramienta, como podemos observar en las imágenes, PMD nos indica línea por línea dónde encuentra algo que comentar y las clasifica por colores para representar la criticidad:

- **Blocker**
- **Critical**

- Urgent
- Important
- Warning

Nos muestra el número de alertas que ha encontrado. En nuestro caso, la mayoría de las violaciones (>800) en el código son consideradas urgentes e importantes, teniendo 65 como bloqueantes o críticas y 85 como warnings.

Tenemos la oportunidad de ver más en detalle cada una de las reglas violadas y mostrar los detalles de la misma. En el ejemplo mostramos una de ellas donde te avisa de que la variable *respuestaSeleccionada* es demasiado larga y te da un ejemplo de uso.

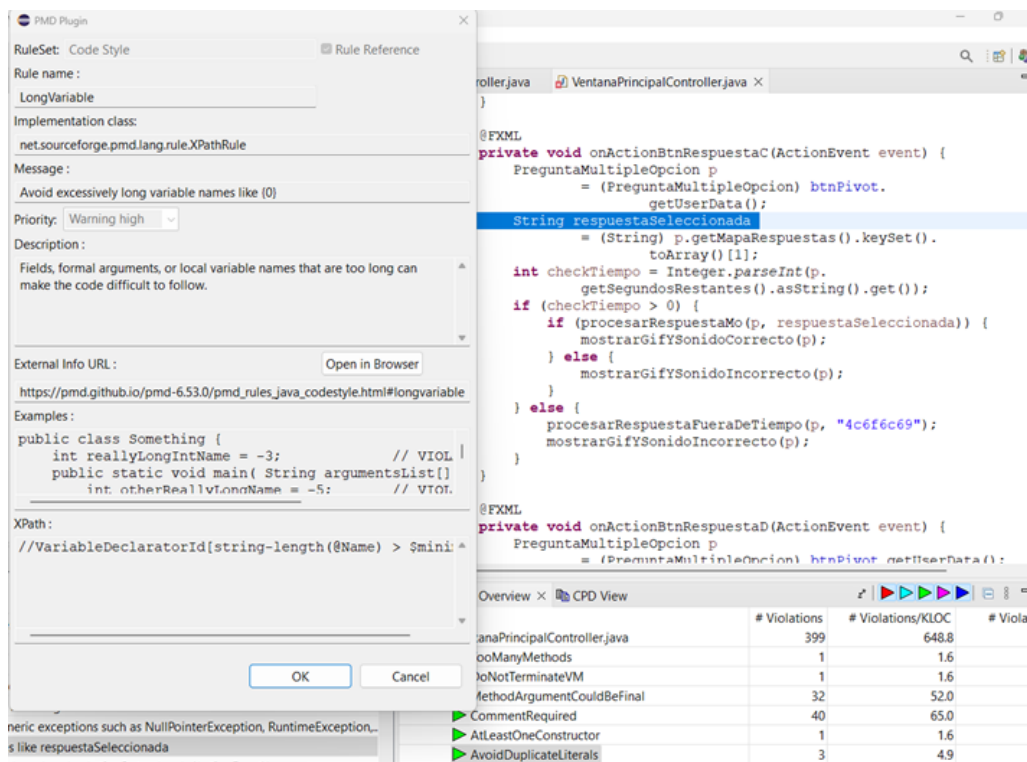


Figure 4.4: Archivos del código fuente

4.5. Valoración de riesgos

Después de hacer estos últimos análisis, valoraremos los siguientes riesgos:

- Documentación en código. El código, a pesar de haber seguido las JCC, carece de explicaciones en las líneas. Si en el futuro un nuevo desarrollador continua con las mejoras y nuevas implementaciones del código, tendrá que dedicar un tiempo para entenderlo y aunque no sea un código extenso, el código debería de estar lo más legible posible.
- Librerías desactualizadas. Al ser un proyecto del 2019 es muy probable que haya librerías que no estén actualizadas.
- Estructura, funciones y variables. Después del análisis estático del código nos da una mejor idea de la cantidad de reglas de programación segura que se salta. Una mala calidad del código puede hacer perder mucho tiempo en encontrar fallos de funcionalidad, es más probable que sea vulnerable frente amenazas y que, además, sea más difícilmente identificable.

4.6. Implementación propuesta

A continuación, como hemos hecho en otras secciones, propondremos una implementación más segura:

- Documentación en código. Como ya hemos mencionado, lo más correcto sería dedicar un tiempo a describir las líneas del código y en caso de que se escribiera más líneas en un futuro, continuar con la documentación. Esta aplicación, al tener un código corto comparado con proyectos que se desarrollan en empresas, no habría problema en repasar las líneas y comentarlas. Dependiendo del nivel de un nuevo desarrollador, esta práctica ayudaría mucho a optimizar el tiempo.
- Librerías desactualizadas. Actualizar las librerías a la última versión estable es posible que arregle ciertos bugs y mejore funcionalidades.
- Estructura, funciones y variables. Gracias a que existen muchas herramientas dedicadas a auditar código fuente y de que te ayuden a seguir las recomendaciones, es más fácil mejorar la calidad del código.

5

Operaciones

5.1. Intro

En este capítulo se busca realizar un análisis y planificación de las operaciones. Esta aplicación, al ser un desarrollo con objetivo meramente educativo, se considera que no existe ningún entorno de producción ni de desarrollo. Tan solo se encuentra en un repositorio de Github.

5.2. Deficiencias

Considerando que en un futuro esta aplicación pueda usarse en instituciones y centros educativos, como ya hemos comentado a lo largo del documento, sería interesante considerar varios entornos para desempeñar funciones diferentes para mejorar la usabilidad de la misma.

- Necesitaremos un primer entorno donde se comience a probar las funcionalidades, esta instancia sería la de **desarrollo**. Aquí, podremos usar la aplicación para hacer pruebas de funcionalidades, de visualización... Se puede manejar en un ordenador o servidor local e incluso, puede ser procesado en herramientas desarrolladas para estas tareas.
- Un entorno de **pruebas** es también interesante para que tanto los propios desarrolladores como los usuarios finales puedan hacer tests de funcionalidades. Pero, de esta fase hablaremos en el siguiente capítulo.
- Antes de pasar al entorno de producción, existe un entorno **pre-producción** el cual simula el entorno final pero donde es posible hacer las ultimas pruebas, como actualizaciones, donde los riesgos son mínimos y es posible revertir la acción fácilmente.
- Y, por último, pero no menos importante, la instancia de **producción**, la cual estará funcionando en un entorno real donde los usuarios podrán hacer uso de la última versión de la aplicación en cuestión.

Al hilo de los comentado, no existe ninguna planificación para los posibles despliegues del juego. Existen muchas plataformas y formas de hacer llegar al usuario la aplicación. Sí que hemos comentado la posibilidad de llevarlo a la web por su facilidad de uso pero no hay pautas ni plan de llevarlo a cabo por el momento.

5.3. Mejoras

Dado el contexto actual, no veo necesario tal despliegue de entornos mencionados en el anterior apartado ya que es una aplicación bastante sencilla. De todas formas, estos entornos sí que serán indispensables en el caso de que se quiera llevar a cabo un desarrollo más extenso de la aplicación.

Para asegurar un despliegue correcto del proyecto a lo largo de los entornos serán necesarias algunas de las pautas que se listan a continuación:

- Auditorías. Indispensable la realización de auditorías periódicamente. Estas auditorías podrán ser tanto internas por los propios desarrolladores, como externas. Se evaluaría dependiendo de la magnitud que vaya obteniendo el proyecto a futuro.
- Monitorización. Este proceso será necesario para poder controlar el avance del proyecto en ejecución. Con esto se evalúa el desempeño de la aplicación en cuanto a rendimiento, capacidad del juego, etc. y los compara con resultados previstos por los desarrolladores.

6

Automatización

6.1. Introducción

Finalmente, en esta última etapa, se quiere estudiar el proceso de automatizado y testeo. Esta automatización de las pruebas permite comparar los resultados obtenidos de los que esperábamos en el funcionamiento de cada elemento de una manera más ágil.

El objetivo de automatizar las pruebas a parte de agilizarlas, se debe también a que, de esta forma, no cueste hacer un testeo de manera periódica.

6.2. Herramientas

En esta parte final del proyecto se realizaron pruebas unitarias. La automatización de las pruebas se llevaron a cabo con la herramienta JUnit 4.12. Para cada una de las clases se realizó una clase de pruebas y para cada método un método de prueba.

También se hizo uso del plugin JaCoCoverage 0.8.2 para facilitar la información de cobertura de las pruebas que se realizaron. Gracias a esta herramienta se pudo comprobar que tan solo el 59% del dominio fue probado, el cual incluye las clases *Pregunta*, *PreguntaCortaRespuesta*, *PreguntaMultipleOpcion*, *PreguntaVF* y *Sistema*. Los archivos *MainApp*, *FXMLController* y *VentanaPrincipalController* tienen un porcentaje de cobertura del 0%.

Obligatorio







Element	Missed Instructions	Cov	Missed Branches	Cov	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.barrios_irabedra.obligatorio.controladores		0%		0%	126	126	443	443	37	37	2	2
com.barrios_irabedra.obligatorio.dominio		59%		41%	49	106	112	285	17	59	0	6
com.barrios_irabedra.obligatorio		0%		n/a	5	5	11	11	5	5	2	2
Total	2.079 of 2.669	22%	221 of 256	13%	180	237	566	739	59	101	4	10

Figure 6.1: Resumen de cobertura de pruebas unitarias

En resumen, el total de nuestra cobertura es de un 22% para la cobertura de instrucciones y de un 13% para las diferentes ramas, estando excesivamente por debajo de lo tolerable siendo 90% el aceptable y 100% el idílico.

6.3. Pruebas funcionales

Además de la automatización de las pruebas, se hicieron también unas pruebas funcionales. Previo a estas

pruebas, se examinaron cuatro etapas en el ciclo de ejecución del juego y dentro de esas etapas se han considerado varios casos de uso. A continuación se detallan los resultados obtenidos:

1. Cargar preguntas desde archivo

-Precondiciones: El usuario debe haber ejecutado el juego.

-Descripción: El usuario debe arrastrar archivos .txt en formato GIFT para cargar las preguntas al quiz.

Table 6.1

Descripción de acciones	Resultado
Arrastrar un archivo .txt en formato GIFT	✓ El sistema indica cuántas preguntas se cargan
Arrastrar un archivo que no es .txt o no tiene formato GIFT	✓ El sistema no actualiza la cantidad de preguntas
Cargar un .txt cuando ya hay 39 preguntas	✓ El sistema no actualiza la cantidad de preguntas
Cargar un .txt de más de 39 preguntas	✓ El sistema recoge las primeras 39 preguntas
Hacer click en 'Comenzar' sin preguntas cargadas	✓ El sistema espera a que se cargue alguna pregunta antes de avanzar
Hacer click en 'Comenzar' con preguntas cargadas	✓ El sistema ingresa al tablero de juego

2. Iniciar quiz

-Precondiciones: El usuario debe haber cargado las preguntas exitosamente y encontrarse en el tablero.

-Descripción: El usuario tiene la opción de seleccionar algún casillero con pregunta cargada para responder a ella. También puede utilizar los botones para acabar, pedir ayuda o generar reporte.

Table 6.2

Descripción de acciones	Resultado
Pulsar el botón 'Ayuda'	✓ El sistema despliega la ventana de ayuda
Pulsar el botón 'Acabar'	✓ La ventana del juego se cierra
Pulsar el botón 'Reporte'	✓ No sucede nada puesto que no se ha respondido a ninguna pregunta
Utilizar el deslizador del volumen	X No sucede nada: no está implementado aún

3. Responder preguntas

-Precondiciones: El usuario debe haber cargado las preguntas exitosamente y encontrarse en el tablero.

-Descripción: El usuario debe elegir algún casillero con pregunta para responder a ella.

Table 6.3

Descripción de acciones	Resultado
Presionar un botón en gris oscuro	✓ No sucede nada. La casilla no está habilitada ya que no contiene pregunta
Presionar un botón gris claro correspondiente a una pregunta V/F	✓ El sistema despliega un panel donde se muestra la pregunta, el tiempo restante y botones de verdadero y falso
Presionar un botón gris claro correspondiente a una pregunta corta	✓ El sistema despliega un panel donde se muestra la pregunta, el tiempo restante, un campo para escribir y un botón 'ok'
Presionar el botón 'ok' de la pregunta corta dentro del tiempo	✓ El sistema no responde y espera a que se ingrese texto en el campo
Presionar un botón gris claro correspondiente a una pregunta múltiple opción	✓ El sistema despliega un panel donde se muestra la pregunta, el tiempo restante y las respuestas posibles
Responder a una pregunta una vez que el tiempo ha terminado	x El sistema indica con un sonido (solo en IDE) y un GIF que la respuesta fue incorrecta y redirige al tablero donde indica con una X el casillero de la pregunta y lo vuelve de color azul oscuro
Responder correctamente antes de que acabe el tiempo	x El sistema indica con un sonido (solo en IDE) y un GIF que es correcta y redirige al mapa de botones, donde indica con un tik el casillero de la pregunta y lo vuelve de color celeste
Responder incorrectamente antes de que acabe el tiempo	x El sistema indica con un sonido (solo en IDE) y un GIF que es incorrecta y redirige al mapa de botones, donde indica con una X el casillero de la pregunta y lo vuelve de color azul oscuro

4. Generar reporte

-Precondiciones: El usuario debe encontrarse en el tablero y haber respondido al menos a una pregunta.

-Descripción: El usuario genera un reporte

Table 6.4

Descripción de acciones	Resultado
Presionar el botón 'reporte'	X El sistema NO genera un reporte aunque exista el archivo 'reporte.pdf'. Este se encuentra en la carpeta del proyecto \Obligatorio\src\main\resources que debería contener las preguntas con las respuestas e información

Estos errores en la funcionalidad ya han sido mencionadas en el capítulo 4 Implementación, sección 4.4 Fallos en el código.

7

Conclusión

El objetivo de este documento ha sido hacer una auditoría de seguridad de la aplicación *LaEspiral-2.0* y proponer posibles mejoras a implementar. Con esta sección se pretende hacer un resumen del análisis y de los resultados que hemos ido comentando a lo largo del reporte, y también poder realizar un planteamiento a futuro.

Hemos podido comprobar que la aplicación carece de recursos que consoliden la seguridad, y aunque no se hayan podido considerar todos los posibles riesgos, sí que hemos podido ver bastantes deficiencias a lo largo del documento. Además de considerar insuficiente uno de los objetivos de las bases de la ciberseguridad, el **desarrollo seguro**, también hemos visto claros errores de funcionalidad que, incluso en un código sencillo, por falta de tiempo no fueron solucionados, haciendo que la **calidad del software** sea también insuficiente.

Bien es cierto que, como acabamos de mencionar, es un desarrollo de un SW bastante sencillo y de uso bastante corriente pero que tampoco no interactúa por el momento con otras máquinas y que no maneja datos personales ni confidenciales, por lo que por ese lado estamos más tranquilos.

No obstante, puede ser una primera versión para poder seguir trabajando en un proyecto mayor. Por ejemplo, la idea de llevar la aplicación a la Web o alguna plataforma de aplicaciones móviles requerirá un despliegue de arquitectura y donde sí que tendremos que tener en consideración lo mencionado en la sección 2 Arquitectura.

Por finalizar, cabe destacar que gracias a este análisis se ha visto la importancia de considerar la parte de seguridad desde las primeras fases de trabajo y la necesidad de aplicar las reglas y recomendaciones. Nuestra herramienta puede tener una idea potente, pero si este corre el riesgo de sufrir un ataque, de que no sea mantenible o que simplemente no tenga buena funcionalidad terminará por reemplazarse por otra mejor opción. Basta con unas buenas pautas considerando los cinco principios básicos de la ciberseguridad recogidos en el libro *Secure coding: principles and practices* y que hemos explicado uno a uno se puede conseguir un SW de calidad.

Appendices

A

Cuestionario



INFORMATION SECURITY REVIEW QUESTIONNAIRE

This questionnaire facilitates the identification of security requirements for a CUNY information technology project, application or system. The questionnaire is intended for those CUNY projects, applications and systems that involve Non-Public University Information or that acquire ongoing vendor IT services (e.g., application software hosting, hardware/software infrastructure, data storage facilities, staffing, etc.)

CUNY/CIS Information Security
<http://security.cuny.edu>
security@cuny.edu

V1.2

Introduction

Identifying information security requirements in the earliest planning stages of a technology project is important to reduce the risk of introducing new security issues into the University environment. Involving CUNY/CIS Information Security early on also minimizes potential project schedule delays when security requirements are retrofitted into systems and/or contractual agreements late in the process.

Name of Application: LaEspiral-2.0

1. DATA CLASSIFICATION

Purpose *This section identifies the highest sensitivity level of data that the project involves. This information is needed to determine baseline data security requirements that must be addressed during the project.*

1.1. The project involves: *(check all that apply)*

☐ Non-Public University Information

Subcategories:

- ☐ Personally Identifiable Information
- ☐ Educational records and/or other information subject to FERPA regulations
- ☐ Information regarding an individual's mental or physical condition and/or history of health services use and/or other information subject to HIPAA regulations
- ☐ Financial information, including credit card and bank information, budgeting, salary and financial aid information
- ☐ Human Resources information
- ☐ Research information
- ☐ Other data the project sponsor considers sensitive, private, confidential or non-public

If any box above is checked, explain the nature, type and quantity of the data and why the involvement of this non-public university data is essential to the system or service to be delivered by the project:

No corresponde a ninguna casilla ya que no contiene información confidencial, datos sensibles ni privados.

2. USE OF VENDOR IT SERVICES

Purpose *This section describes the intent, if any, to acquire ongoing vendor IT services (e.g., application software hosting, hardware/software infrastructure, data storage facilities, staffing, etc.) in support of this project or service. This information is needed to determine security requirements that should be considered when evaluating vendor services and negotiating vendor contracts.*

2.1 Will the project acquire ongoing vendor IT services (e.g., application software hosting, hardware/software infrastructure, data storage facilities, staffing, etc)?

- ☒ Yes
- ☐ No. If checked, skip to Section 3.

2.2 The vendor service(s) will be acquired via:

- ☒ Request For Proposal
- ☐ Sole Source Procurement
- ☐ Purchase Order
- ☒ Agreement to vendor's online license user agreement
- ☐ Other. If checked, describe here:

[Click here to enter text.](#)

2.3 Briefly describe below the service(s) to be acquired, including names of desired vendor(s) if known:

La aplicación puede ser que se suba a la web mediante alguna compra de dominio o que se aloje en la nube. Se puede proponer a un vendedor el Proyecto o comprarlo directamente nosotros, aunque suponga un gasto en el mantenimiento.

3. Identity Management, Access Control, Authorization

Purpose

This section identifies the user population who will have access to the IT product or service to be delivered by the project, as well as planned security access controls. This information will help determine if additional controls are needed to reduce the risk of unauthorized or otherwise inappropriate access to sensitive data.

3.1. Who will access this application or system?

- ☒ Faculty
- ☐ Staff
- ☒ Students
- ☐ Consultants and temporary employees
- ☐ Other (please explain):

Esta aplicación esta pensada para estudiantes y centros educativos.

3.2. If not covered above, what entities external to the University will have access to the application or service? Cualwuier usuario puede acceder al juego.

3.3. Is access limited to only those individuals whose job or function requires such access? No

3.4. Is any part of the system open to the public or to an anonymous class of users? Está actualmente abierta al público

3.5. Briefly describe the process by which authorization of users will likely be accomplished, if known. No hay ningún proceso de autenticación aunque en el future se quiera implementar.

- 3.6. Are there different levels of authorization in the system? (e.g., full access, limited access, read-only access, etc.) **No. No hay control de accesos en ningún caso**
- 3.7. Is there an identified authority that approves requests for access to this system? Who would that be? **No.**
- 3.8. Is there a process for the access administrator to be notified when a user's status or role changes? **No, no hay Sistema de autenticación**
- 3.9. Will there be uniquely identifiable accounts for all users requiring access? [Click here to enter text.](#)
- 3.10. How will accounts which are no longer needed be recognized and deleted in timely and manageable manner? **En el future cuando exista el control de acceso, habría que plantear un proceso de borrado o bloqueo de usuarios cuando se lleve más de x tiempo sin acceder**
- 3.11. How will this system authenticate users?
- ☐ CUNY Portal LDAP Single-Sign On
 - ☐ Active Directory (cuny.adlan)
 - ☐ CUNY Enterprise Active Directory
 - ☐ CUNYfirst Single-Sign On
 - ☒ Local Authentication
 - ☐ Other:
- [Click here to enter text.](#)
- 3.12. Where local authentication is used, provide details on the enforced password complexity and expiration policy. **No aplica**
- 3.13. Where local authentication is used, provide details on how passwords are securely stored within the system (e.g., encrypted using a salted hash). **No aplica**
- 3.14. Does the application automatically log off, lock or terminate a session after a predetermined time of inactivity? Provide details. **No aplica**

Network Access and Communication**Purpose**

This section identifies the scope of network access requirements. This information will help determine controls needed to reduce the risk of unauthorized or otherwise inappropriate access to sensitive data.

3.15. Is this system required to be network accessible? ☐ yes ☒ no

3.16. If so, will it be accessible:

- ☐ only within CUNY Central Office networks
- ☐ only within one or more CUNY Campus networks (specify)
- ☐ both CUNY Central Office and CUNY Campus networks
- ☐ the Internet at large
- ☐ other – please explain:

No se necesita conexión a internet.

3.17. If available, provide a network diagram that depicts required connectivity for all of the components of the application or service.

3.18. Will this system be accessible through means other than the network (e.g., telephone)? Los planes a futuro son la subida a la Web pero de momento solo es accessible localmente.

4. Data Protection**Purpose**

This section identifies available data protections and requirements. This information will help determine controls needed to reduce the risk of unauthorized or otherwise inappropriate access to sensitive data.

4.1. Are there restrictions on what quantity or type of data can leave the system? Please explain.

No hay restricción

4.2. Are shadow copies of any of the data anticipated to be created? For example, would users copy or download data to their own devices? If so, please explain. No

4.3. Does data associated with this application or system interface with other applications or systems? If so, please provide details. **No aplica**

4.4. Is non-public university data encrypted while at rest? **No aplica**

4.5. Is the data encrypted while transmitted over an untrusted network? **No aplica**

4.6. What type of encryption is used? How is it configured and deployed? **No aplica**

5. Logging and Auditing

Purpose

This section identifies available activity logging and auditing capability. This information will help determine whether additional logging and auditing features needs to be established.

5.1. Describe logs and/or audit trails that are produced by the application or service. **No aplica**

5.2. Is sensitive data embedded in the logs? **No**

5.3. Can logs and/or audit trails link actions to individual users? **No aplica**

5.4. Are successful/unsuccessful accesses logged? With client network address? **No aplica**

5.5. For how long are logs retained? **No aplica**

6. Business Continuity / Disaster Recovery**Purpose**

This section identifies business continuity and disaster recovery provisions and requirements.

- 6.1. Is there a documented business continuity / disaster recovery plan that addresses procedures to restore any lost data or functionality in the event of an emergency or other occurrence, the staff responsible for carrying out data restoration, emergency contact names and numbers, important business partners and other business supply information necessary for a temporary office setup to support data restoration?

No, actualmente no hay un servicio de incidentes

7. Other Comments

A pesar de no haber un log de los mensajes del Sistema, existe un reporte que el usuario deberá de generar primeramente para ver el historia del respuestas dadas en el juego

Diagrama de clases

