15-12-2017

Zaret Roque – Luis Galicia

Universidad veracruzana

Cinco en línea – Entrega final

Tecnologías para la construcción de software

Contenido

[Introducción 2](#_Toc501048558)

[Repositorio 2](#_Toc501048559)

[Commits 2](#_Toc501048560)

[Especificación de requerimientos 3](#_Toc501048561)

[Selección del framework de desarrollo 3](#_Toc501048562)

[Documentación del código 3](#_Toc501048563)

[Funcionalidad 4](#_Toc501048564)

[Pruebas unitarias 4](#_Toc501048565)

[Manejo seguro de datos 4](#_Toc501048566)

[Calidad del código 4](#_Toc501048567)

[Análisis estático de código 4](#_Toc501048568)

[Conclusiones 5](#_Toc501048569)

[Lecciones aprendidas 5](#_Toc501048570)

[Zaret 5](#_Toc501048571)

[Luis 5](#_Toc501048572)

# Introducción

Este documento tiene como propósito integrar todos los artefactos utilizados durante el desarrollo del producto de software Cinco en línea. El producto es un juego en red multijugador, permite el registro de nuevos usuarios, inicio de sesión entre otras funcionalidades que se pueden ver con mayor detalle en la Especificación de Requerimientos de Software del sistema.

Este producto es el resultado de lo aprendido a lo largo del semestre en Tecnologías para la construcción de software, integra diversas prácticas que fueron realizadas a lo largo del semestre, además de conocimientos adquiridos en otras experiencias educativas previas.

# Repositorio

Para trabajar a lo largo del semestre se creó un repositorio en Git, donde se fueron almacenando todos los artefactos y los cambios que se realizaron sobre los mismos. El repositorio está disponible en el siguiente vínculo: <https://github.com/ZaretRoque/TCS>

## Commits

Durante el desarrollo se dividió la carga de trabajo entre los dos integrantes del equipo y ambos realizaron un número similar de commits en el repositorio. En la siguiente imagen se puede ver el reporte generado por Git.

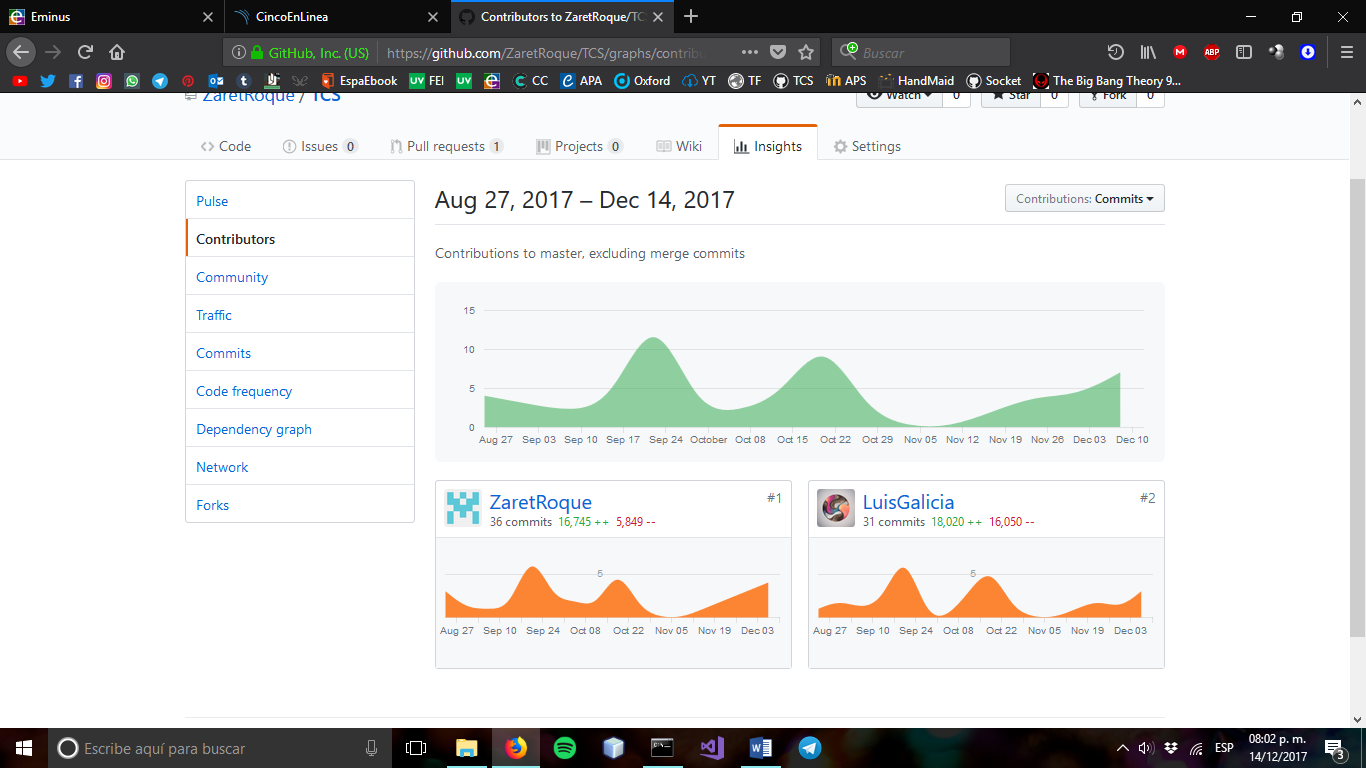


Ilustración 1 Reporte de commits generado por Git

# Especificación de requerimientos

Para el proyecto se realizó una especificación de requerimientos de software, misma que puede ser consultada en el repositorio del proyecto.

# Selección del framework de desarrollo

Elegir un framework para este proyecto fue algo medianamente sencillo. El primer problema al cual nos enfrentamos fue la comunicación con la base de datos, sabíamos que no necesitábamos un framework tan potente, sino uno que nos permitiera el acceso remoto a datos. Para este punto, ya habíamos tenido experiencia con RMI en Java, ahora sólo requeríamos ese conocimiento adaptado a C# (lenguaje en el que se desarrolló este juego). Encontramos el Framework WCF (Windows Communication Foundation), que funciona con la arquitectura cliente-servidor. Nos permitió dar servicios a los clientes conectados, en este caso, consultas a la base de datos, la cual está elaborada en MySql y WCF se adapta al manejador estándar de MySql en C# por lo que no fue necesario descargar ningún archivo. WCF tiene bastante documentación y encontramos muchos ejemplos en la red, así que la curva de aprendizaje para este framework fue relativamente baja, ya que logramos aprenderlo en menos tiempo del esperado. Además del tiempo de aprendizaje, la familiaridad que teníamos al trabajar métodos remotos, la fácil adaptación del framework y toda la documentación encontrada. WCF brinda seguridad en sus protocolos, ya que lo hace por medio de http, puede enviar datos en una dirección (la utilizada para el proyecto) o ambas e incluso mandar la información cifrada, esto último no se hizo para el proyecto por falta de tiempo.

El siguiente problema fue decidir cómo íbamos a implementar la comunicación para el juego en tiempo real, si bien WCF es una alternativa bastante potente, no ofrece las ventajas y características que buscábamos para trabajar. Es por eso que decidimos utilizar Socket.io y Node, en conjunto nos ofrecen la posibilidad de mantener una comunicación en tiempo real mediante un servidor escrito en JavaScript y una implementación del cliente para C#. Socket.io es un framework que funciona dirigido por eventos, por lo cual fue relativamente sencillo aprender a utilizarlo. El framework se encarga de toda la conexión y reconexión de los clientes con el servidor, por lo cual no es necesario preocuparse demasiado por estos aspectos y podemos centrarnos más en la lógica del juego.

En conjunto los dos frameworks nos ofrecen la posibilidad de implementar el juego de una forma sencilla y no hubo problemas al integrarlos. De esta forma exploramos las ventajas de ambos y las utilizamos a nuestro favor, además de reducir la complejidad del proyecto.

# Documentación del código

Los métodos del código están documentados en el estándar para .NET, se describe brevemente el método, los parámetros que recibe y el retorno. De esta forma se garantiza que se entienda la funcionalidad de los métodos y se tenga la suficiente información al utilizarlo sin que sea necesario ver su implementación interna.

# Funcionalidad

El proyecto se encuentra funcionando correctamente y como se espera que lo haga.

## Pruebas unitarias

Para garantizar la funcionalidad se realizaron pruebas unitarias en la capa de datos, específicamente a la clase Consultas, responsable de la comunicación con la base de datos. Y en la lógica del juego, en la clase Crupier, que toma su nombre por la persona que lleva el control en un juego de Póker.

# Manejo seguro de datos

Para el manejo de datos se utilizó el algoritmo de cifrado SHA2 para almacenar las contraseñas en la base de datos, además se utilizan consultas parametrizadas y se controla la entrada de datos en el inicio de sesión y en el registro de usuarios. Lo anterior con el fin de reducir, aún más, las vulnerabilidades del juego ante una inyección SQL. En cuanto a la base de datos, se utiliza un usuario especifico para el juego y que tiene permisos limitados para acceder a la misma.

# Calidad del código

El código está apegado al estándar definido por el equipo, el cual considera mejores prácticas básicas, así como el uso de prefijos para los elementos gráficos. Dicho estándar puede ser consultado en el repositorio del proyecto.

# Análisis estático de código

Para asegurar que el proyecto es de alta calidad se realizó un análisis estático de código a la mitad del desarrollo y al finalizar el mismo, en la siguiente imagen se pueden ver los resultados del análisis.

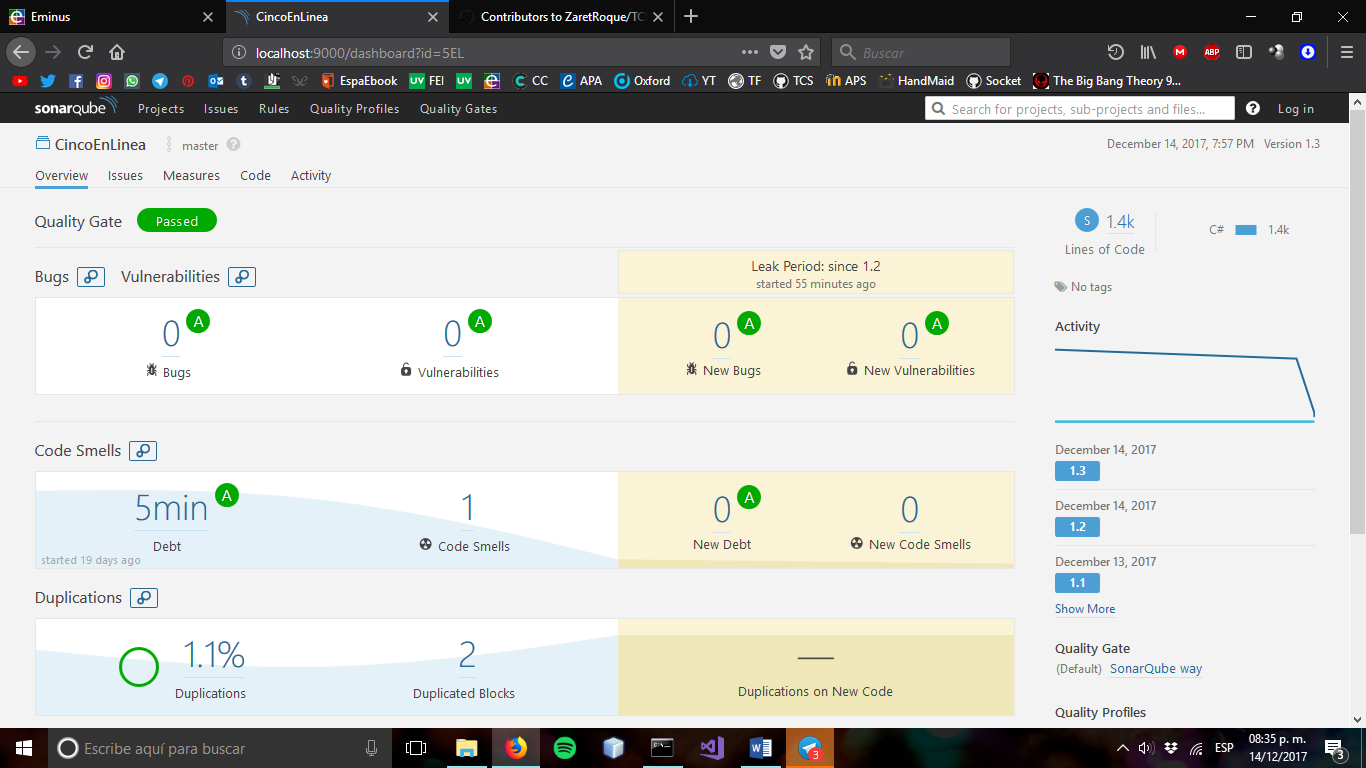


Ilustración 2 Resultados del análisis estático

El análisis se realizó con SonarQube, en la imagen se aprecia una mala práctica, se trata de un bloque *catch* vacío, se programó de esta forma porque es una excepción que puede ocurrir al momento de regresar al Menú principal después de estar en la Sala de espera. Al salir un usuario de la Sala, si este tenía una partida creada, la partida será eliminada, lo cual provoca que se actualice la tabla de partidas disponibles en todos los clientes conectados. Pero al cerrar la ventana, el cliente que regresa al menú no puede ejecutar la actualización, y es donde ocurre la excepción. Fuera de ello el proyecto no tiene vulnerabilidades ni malas prácticas graves.

# Conclusiones

Como equipo creemos que esta experiencia educativa nos hizo aprender y crecer mucho como desarrolladores de software, después de un semestre somos capaces de aplicar frameworks en el desarrollo y crear un sistema distribuido en una tecnología desconocida para ambos, lo que sin duda incrementa nuestra confianza como ingenieros de software. Evidentemente fue un reto pues estábamos acostumbrado a trabajar en Java, pero al ver el trabajo terminado vemos que es posible aprender y crear algo de calidad al mismo tiempo.

No sólo estamos contentos con los resultados del proyecto, estamos seguros que aprendimos muchas cosas útiles, pero también rescatamos puntos como la organización en los tiempos, la comunicación, saber qué tipo de preguntas hacer, aprendimos a explicar nuestro código, compartir ideas para llegar más rápido a la solución y lo más difícil, tomar decisiones rápidas para continuar con el desarrollo. Estamos seguros que esta experiencia educativa ha sido una a la que más jugo le hemos sacado aunque no sabemos si es eso o somos nosotros los exprimidos, sea cual fuese, el conocimiento que adquirimos aquí lo tendremos presente para las futuras experiencias en desarrollo de software que tengamos.

## Lecciones aprendidas

### Zaret

Durante el desarrollo de este proyecto aprendí mucho en lo que se refiere a desarrollo de software distribuido, pero el aprendizaje más fuerte sin duda fue el lenguaje C# y el framework .NET, hacer un cambio de tecnología fue incómodo al principio, puesto que tuve que adaptarme a un nuevo IDE, pero estoy satisfecho con lo que logramos con este proyecto. Me llevo conocimientos para desarrollar con diferentes frameworks, como JPA, WCF, RMI, Socket.io. A pesar de no haber utilizado un framework de persistencia para nuestro proyecto me llevo un poco de experiencia en el uso de JPA y aprendí de sus ventajas y desventajas. También aprendí a tomar en cuenta los hilos al programar una interfaz gráfica de usuario. Sin duda fue un semestre de mucho aprendizaje y me siento preparado para enfrentar los nuevos retos que puedan venir al desarrollar software en el futuro.

### Luis

Desde un inicio, este proyecto representaba un reto, ya que todo lo visto era completamente nuevo para mí. No sólo me enfrenté al problema del autoaprendizaje, que en el tiempo que llevo en la carrera me he dado cuenta de lo fundamental que es, sino también a los problemas colectivos, la organización y los tiempos. Sin embargo, no todo fue malo, me motivaba la idea de desarrollar un juego que pudiera comunicarse en red e ir viendo el proceso poco a poco cambió mi mentalidad. Me di cuenta de todas las herramientas y etapas que lleva la construcción de software a baja escala, además de los recursos para encontrar información útil acerca de los frameworks que utilizamos, en nuestro caso WCF y Socket.io, que en un principio parecía más que una ayuda, un retroceso, pero con los días y la práctica logramos resultados que ni nosotros esperábamos. Nos dimos cuenta de la importancia de preguntar y compartir lo que pensamos, ya que así resolvimos muchos problemas sencillos. Por último, me queda una buena experiencia y ganas de seguir aprendiendo otras tecnologías por mi cuenta, el manejo de datos en red me pareció de lo más interesante, así que continuaré mejorando el juego o tal vez encuentre cosas nuevas en el camino, como me pasó al desarrollar este proyecto, el problema es que por falta de tiempo no pude agregarle todas las monerías que quería.