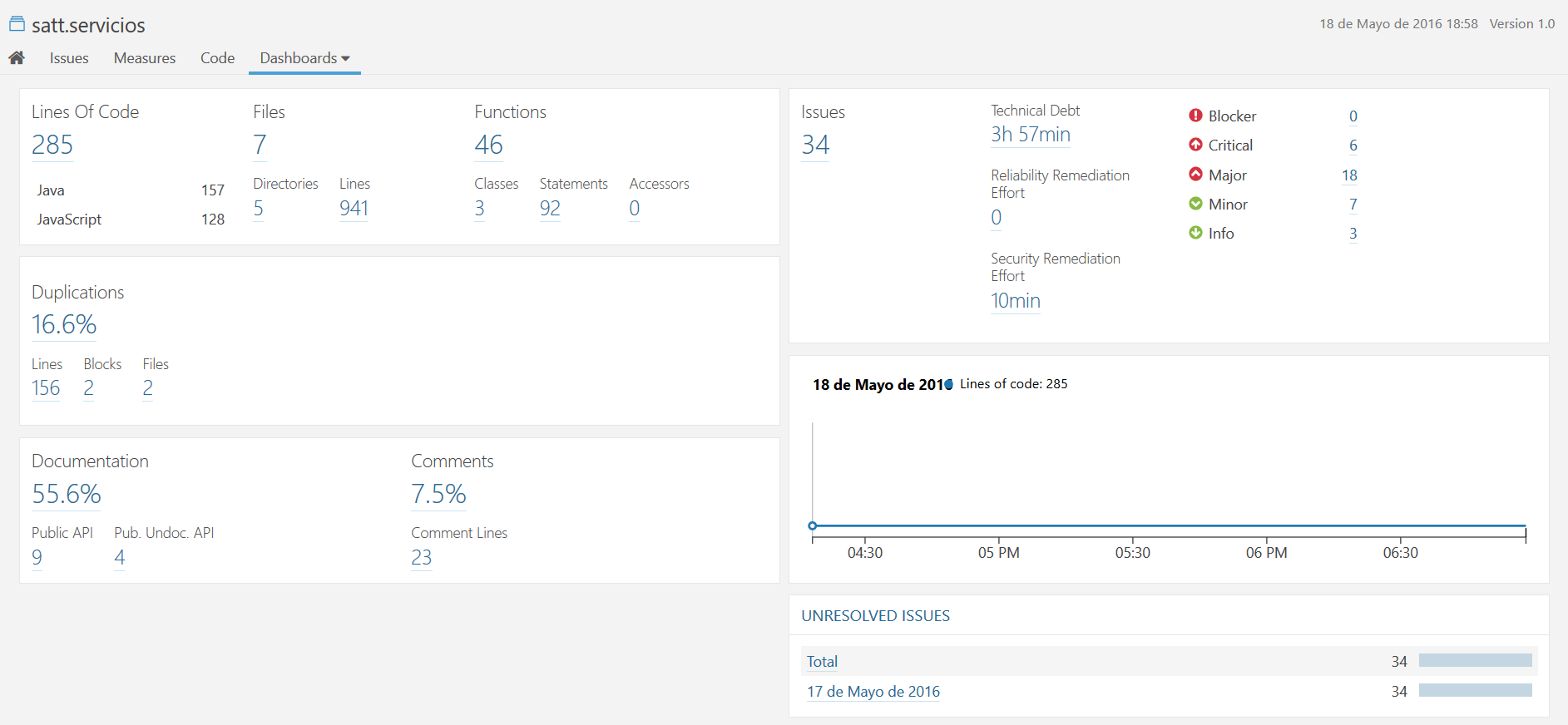
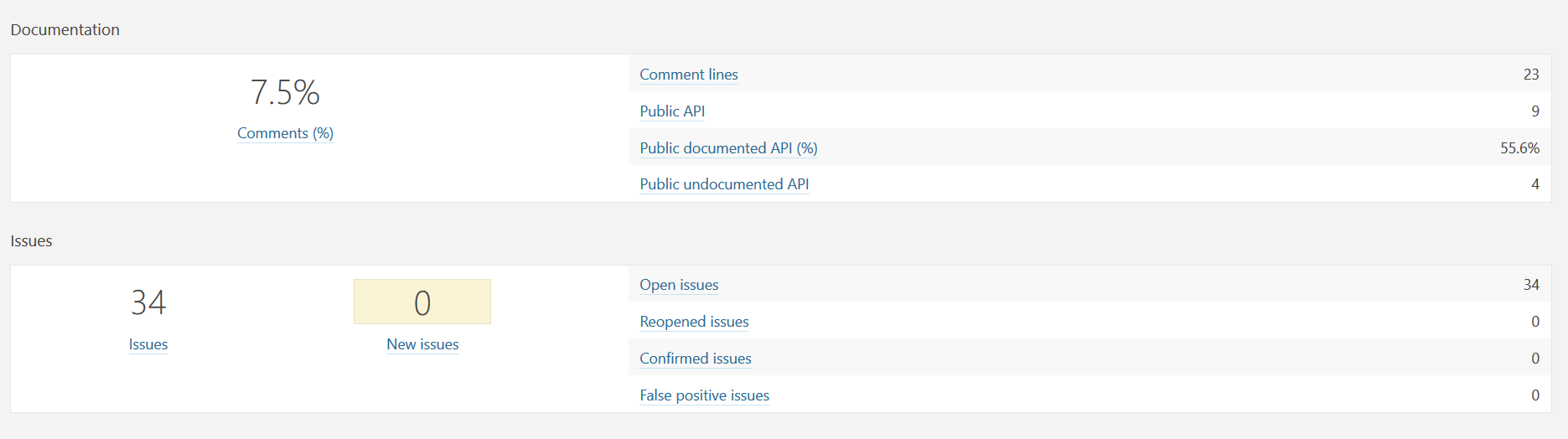
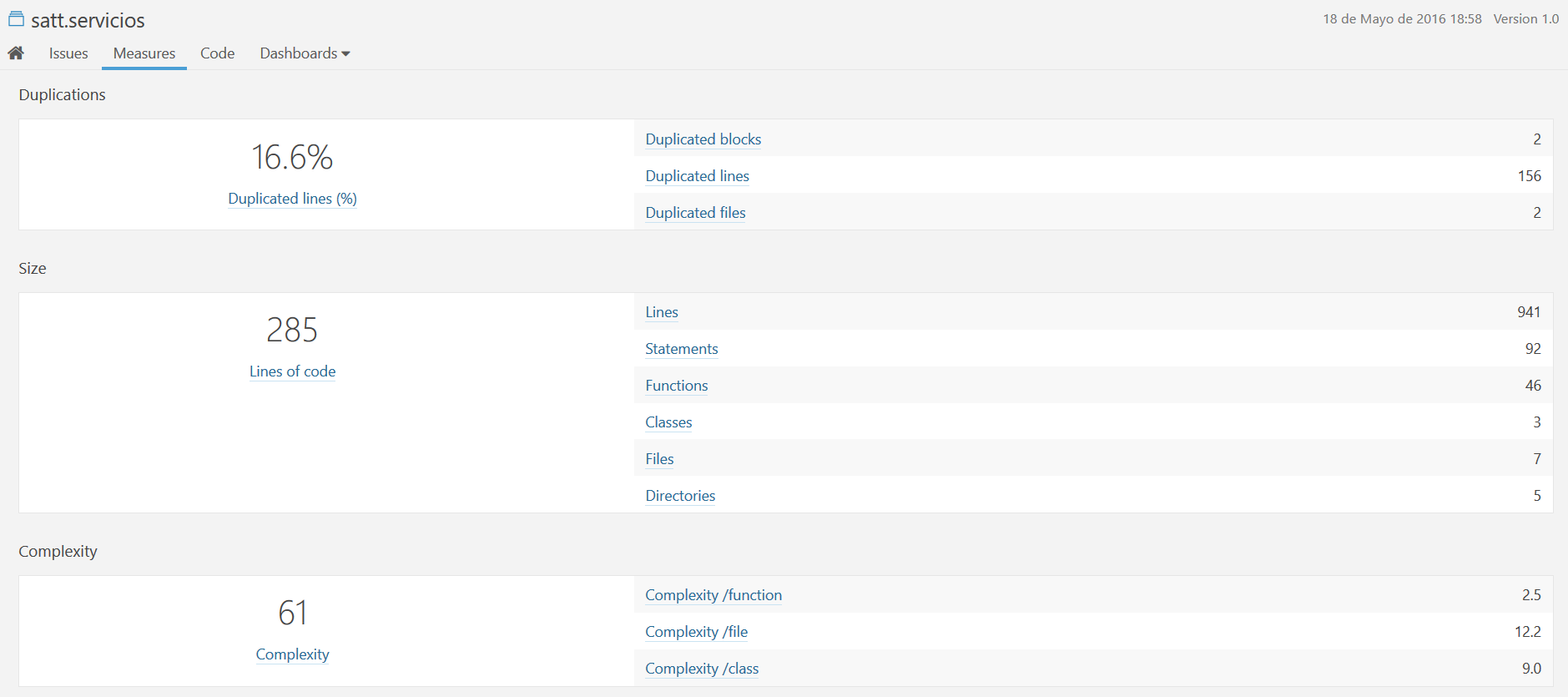
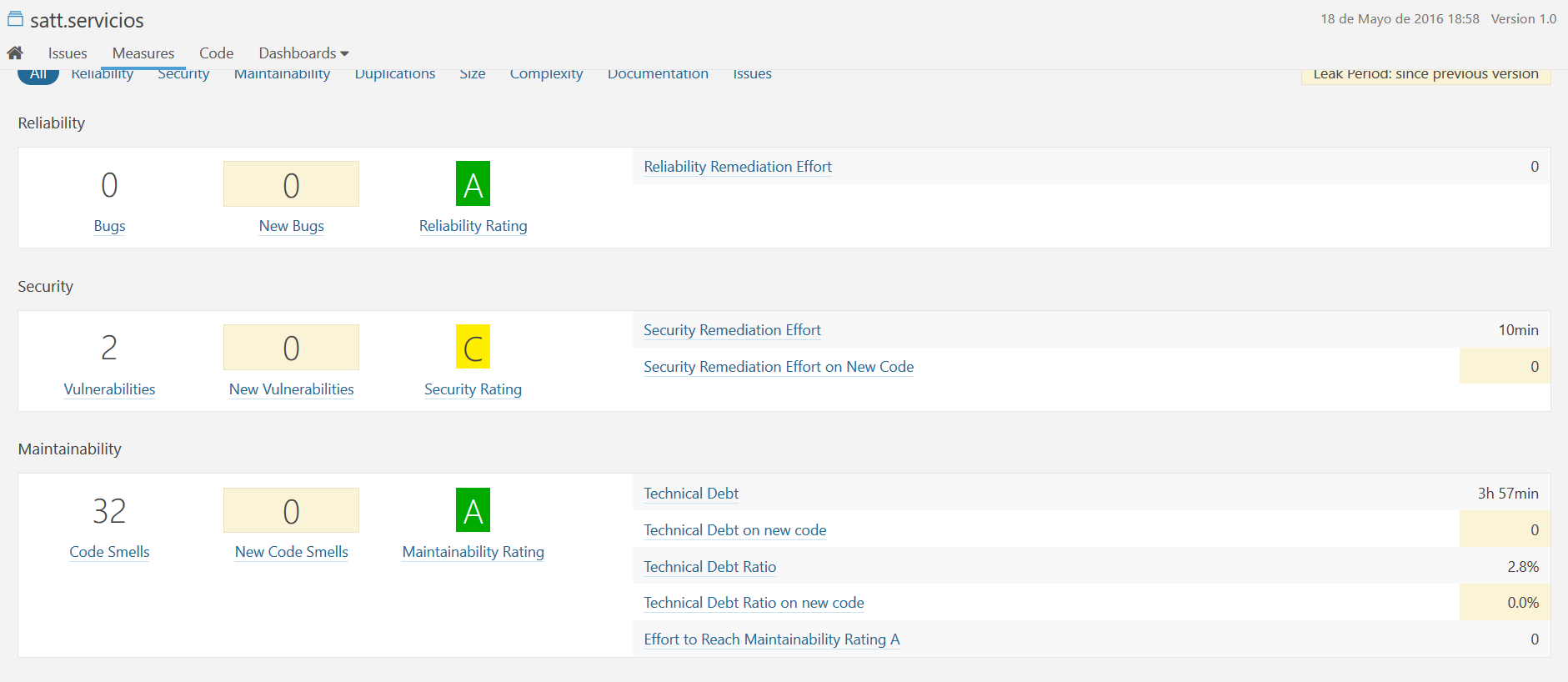
Pruebas Sonar

Se realizaron estas pruebas para las tres partes de nuestro proyecto: interfaz, servicios y backend.

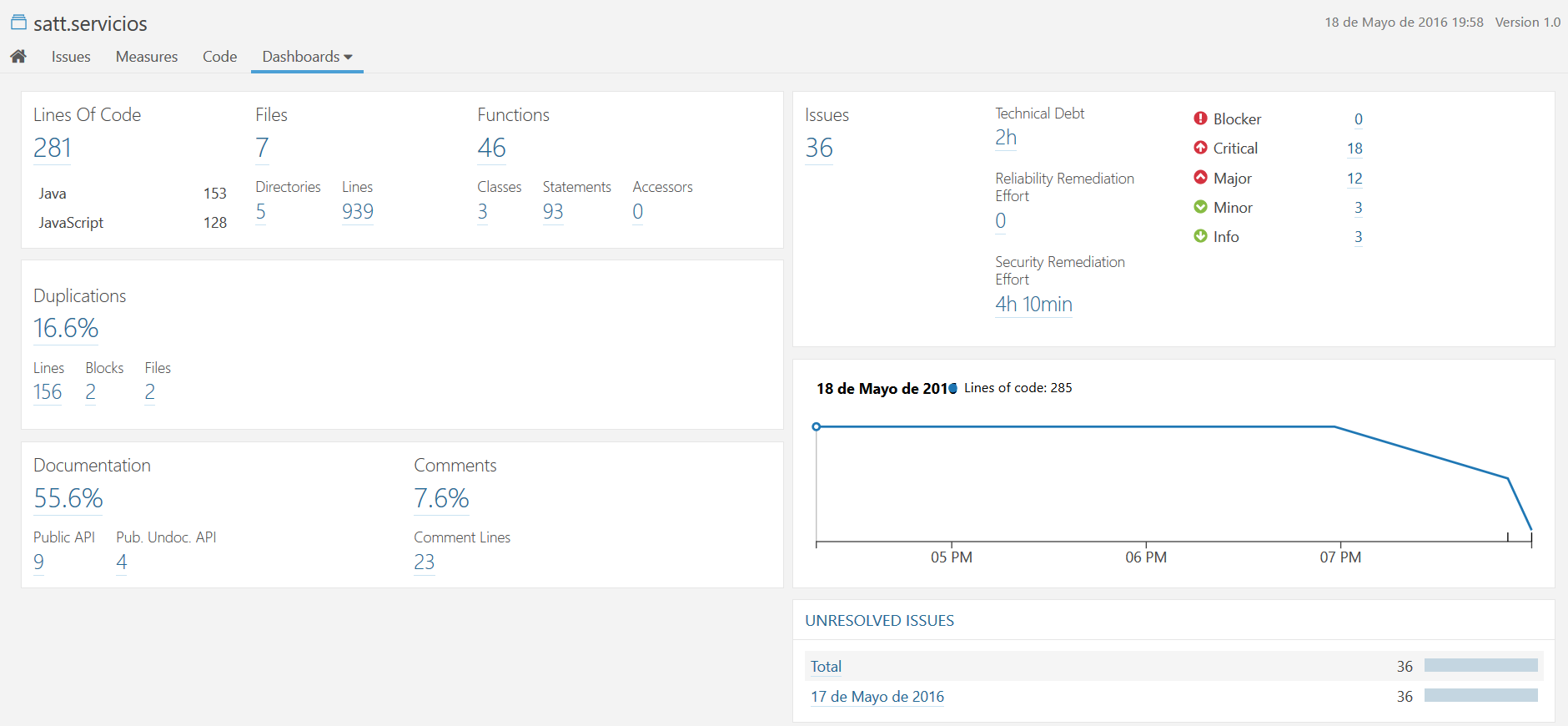
Para la parte de servicios los resultados fueron:





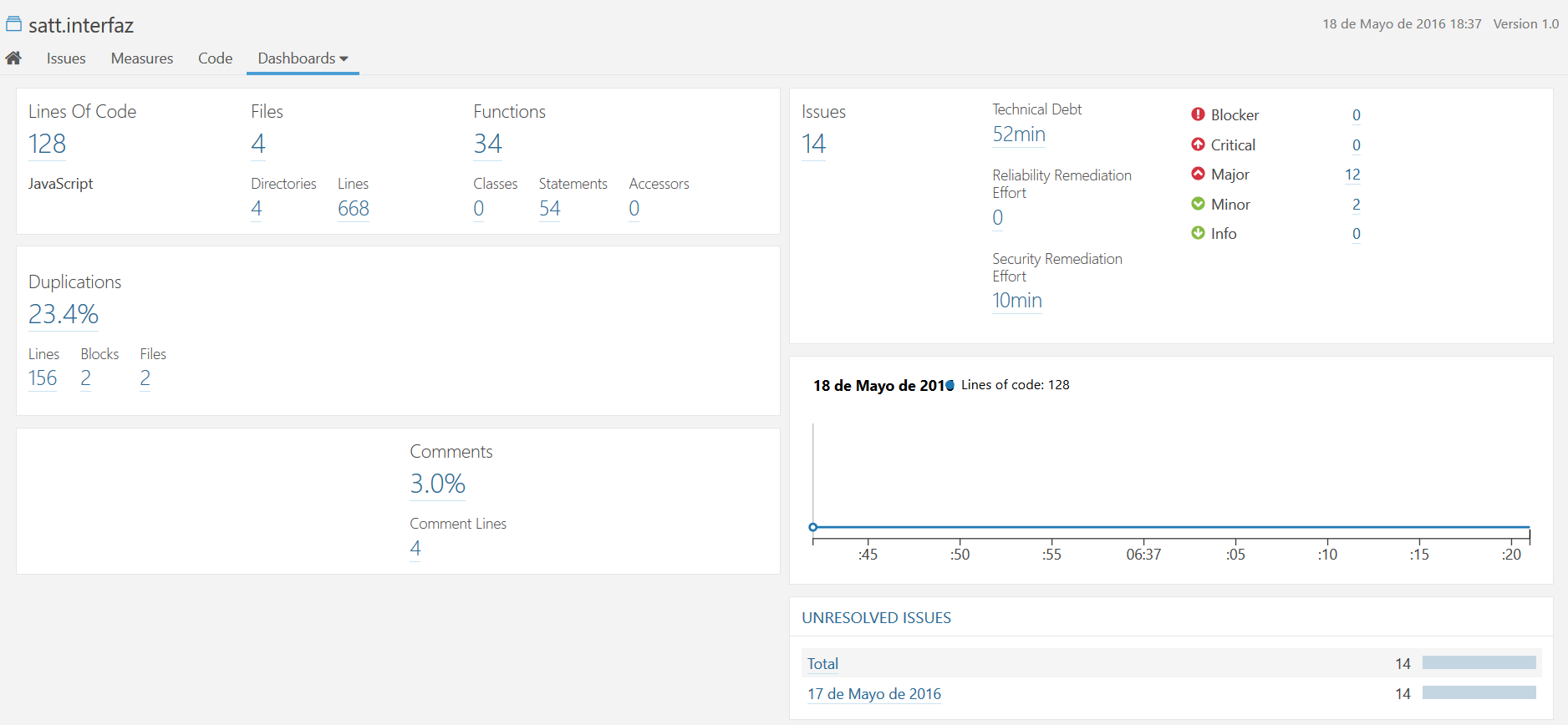
Con estos resultados cabe resaltar los siguientes aspectos: se ve que en principio se generan 34 “issues” de los cuales 6 son críticos y 18 son mayores. La complejidad es muy alta (61), y la seguridad tiene fallas.

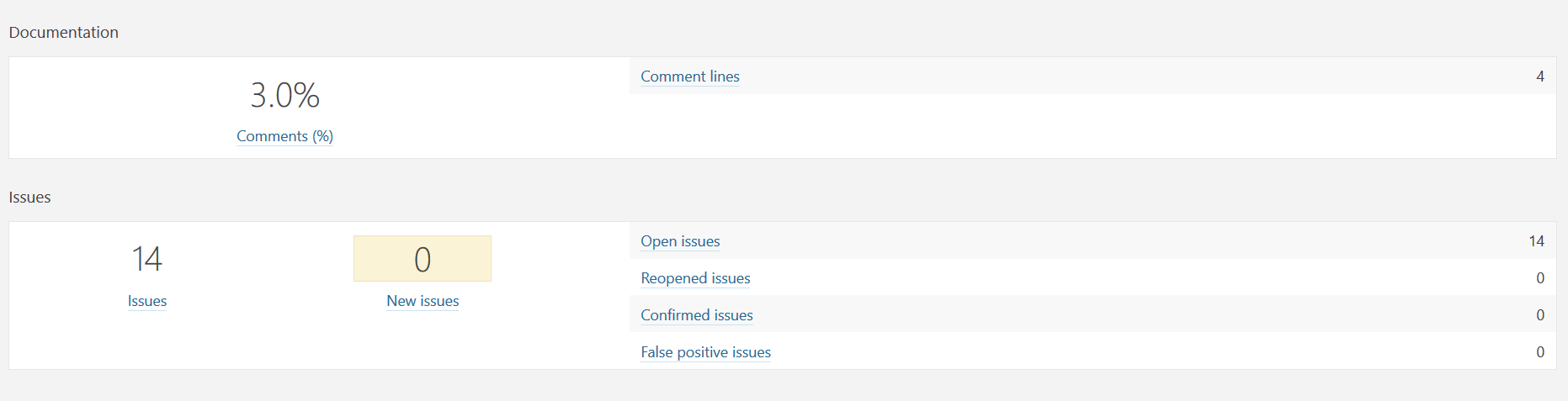
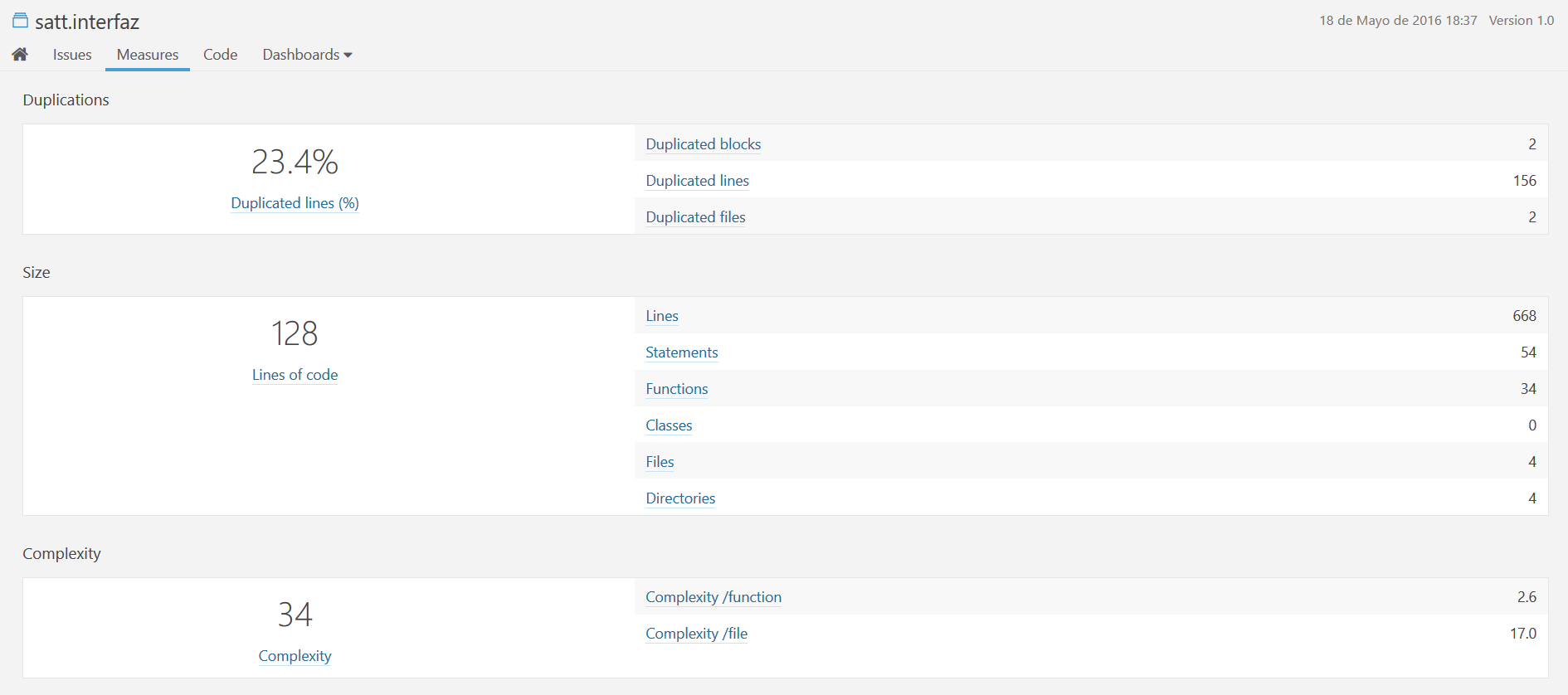
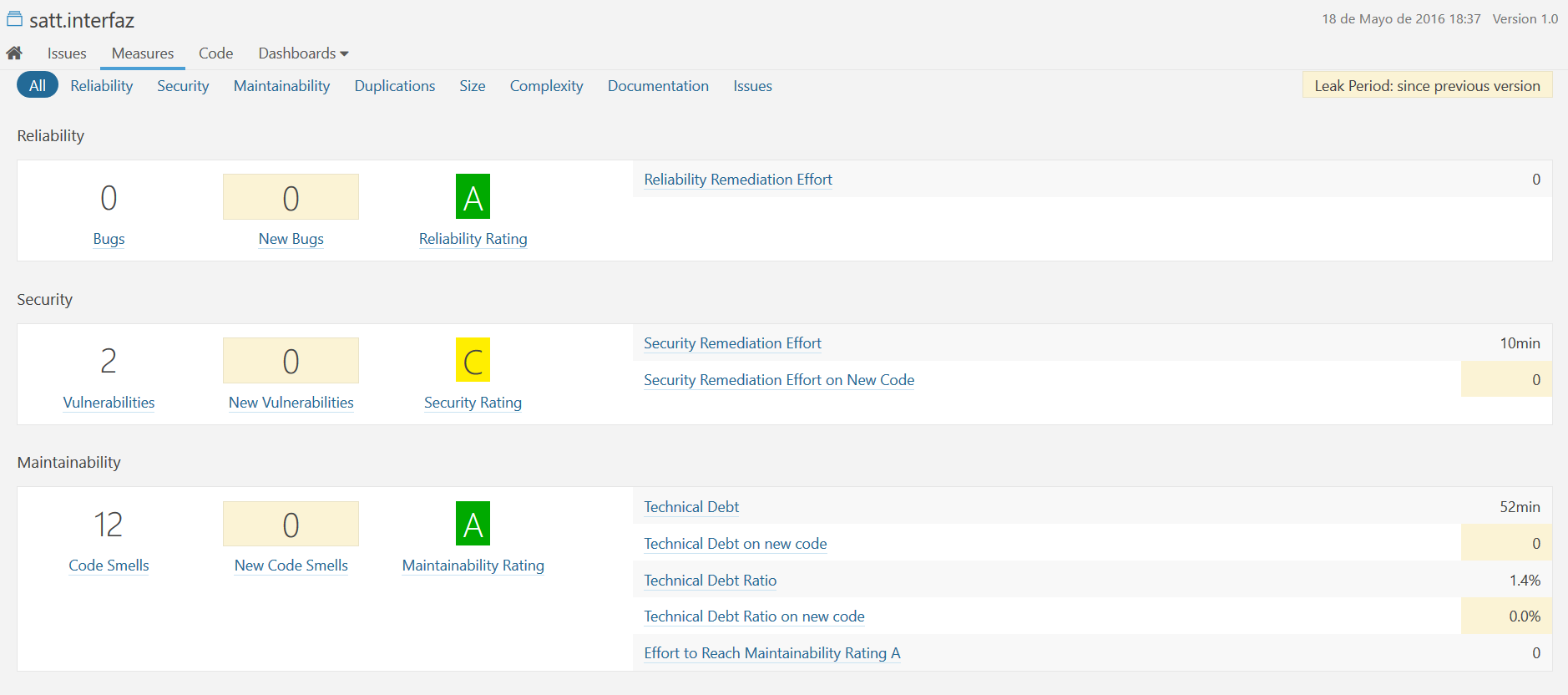
Se intentaron arreglar los “issues” sin embargo el resultado no fue el esperado, ya que solo disminuyeron aquellos de menor importancia y aumentaron los críticos y mayores:



Esto conllevó a que, ya que esta parte de la aplicación no tenía mayor complicación a nivel de código y tenía pocas clases, solo se hicieran cambios menores al código en cuanto imports innecesarios y repeticiones de código. Con esto se mejoraron en pequeña cantidad las estadísticas de un comienzo.

Para la parte de interfaz:

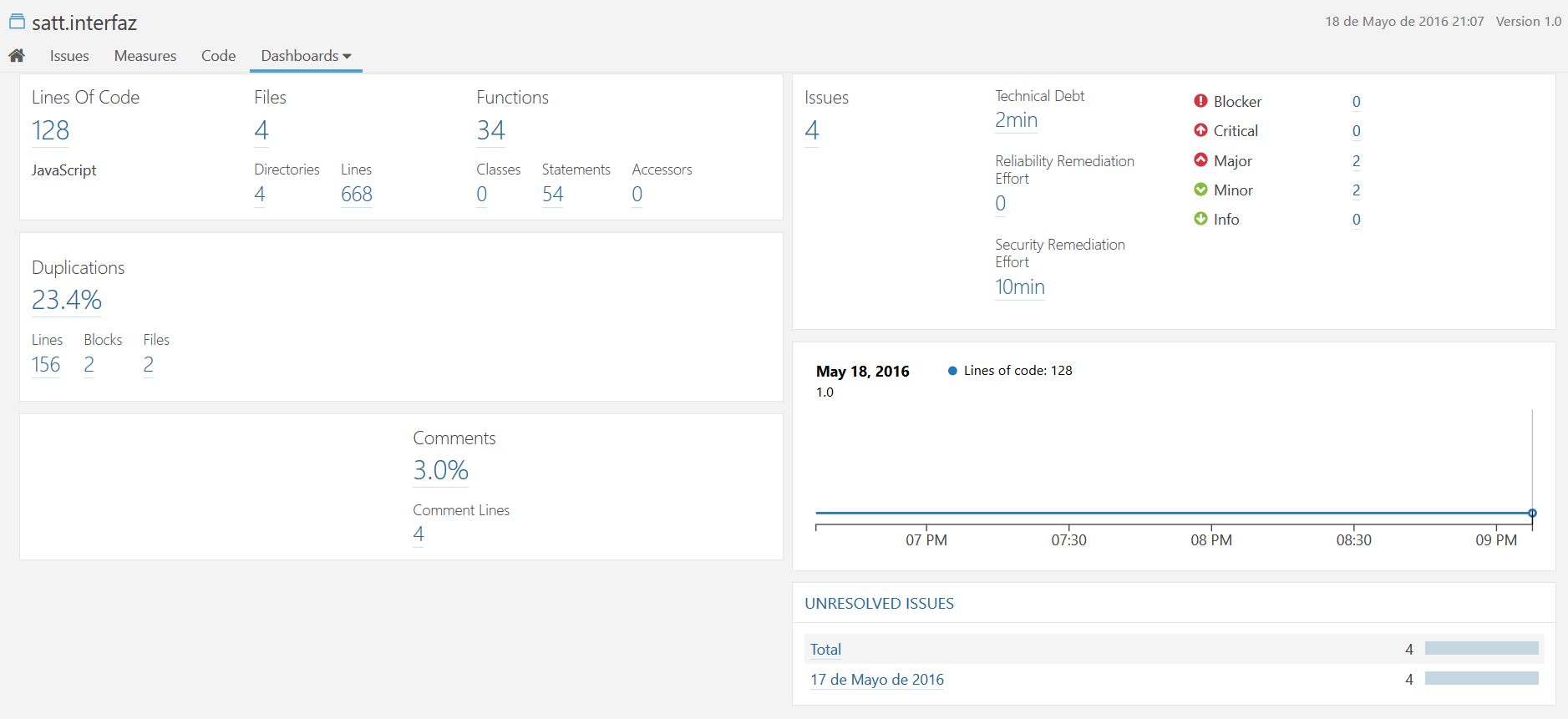




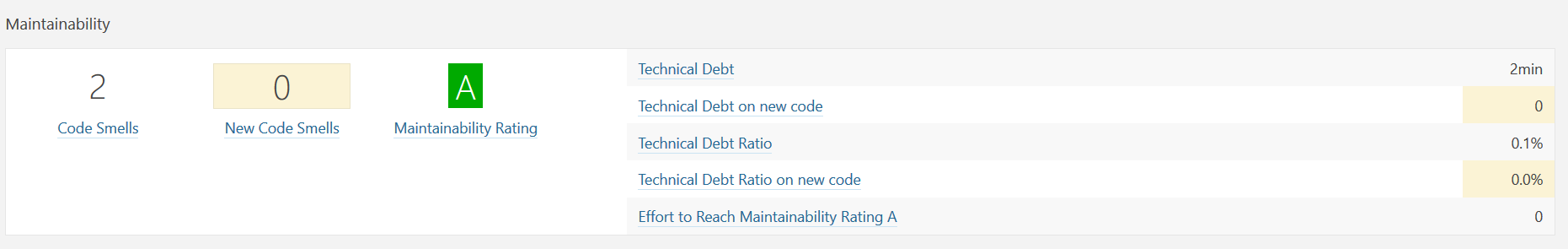
Acá se puede ver que se obtuvo mejores medidas con respecto a la parte de servicios. Solo se tuvieron 14 “issues”, de los cuales solo hay 12 mayores y no hay críticos. En cuanto a la complejidad vemos que también es menor (34).

Teniendo en cuenta los resultados, se intentó hacer un código más modificable, atacando sobre todo los “issues” presentados.

Los resultados fueron:

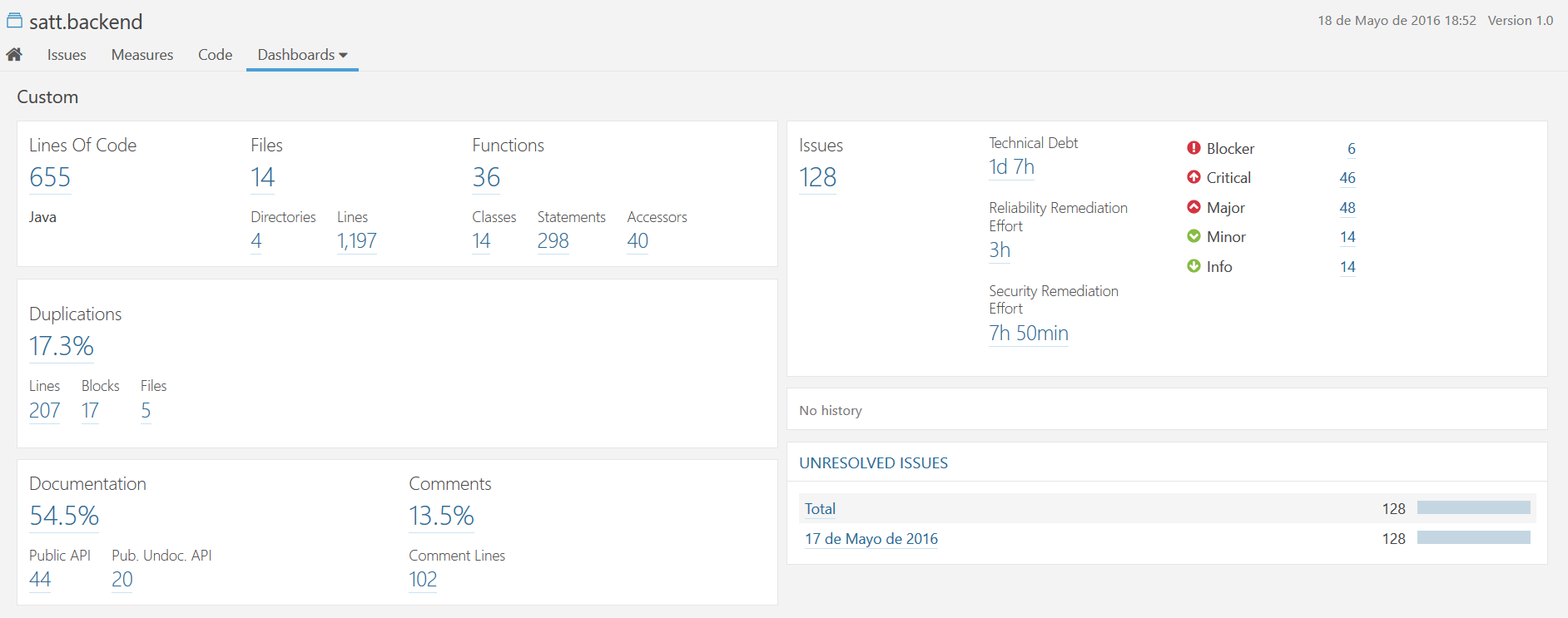


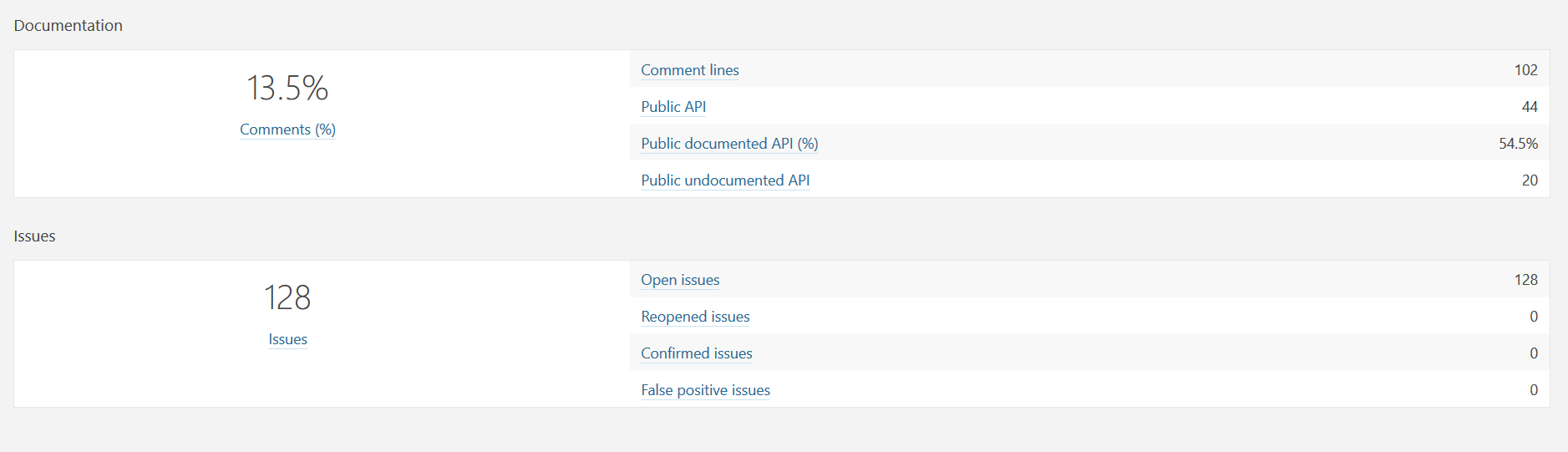
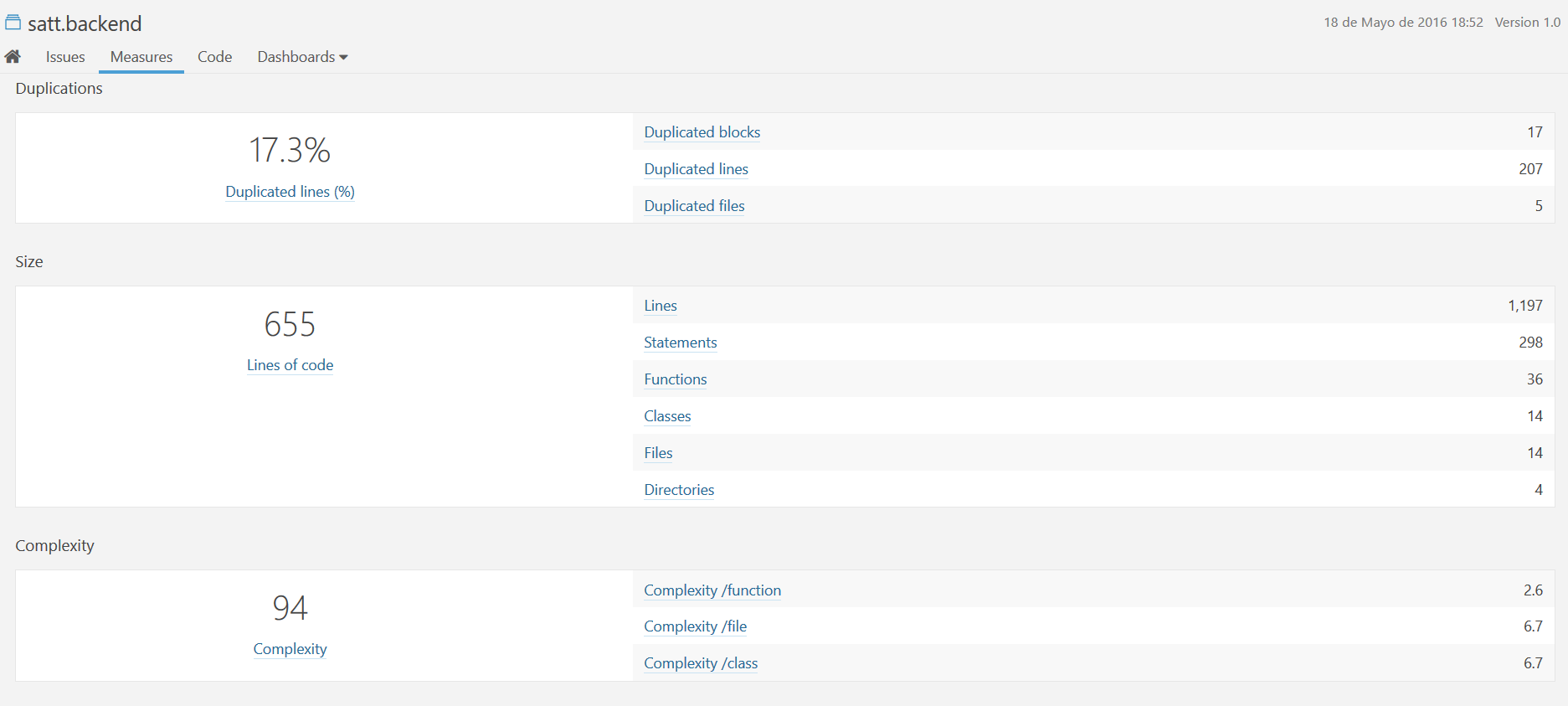
Como se puede ver, al modificar el código, se mejoró la modificabilidad. Los issues se redujeron drásticamente a solo dos mayores y dos menores. Se puede ver que hubo una mejoría muy notoria en la cantidad de “code smells” como se muestra a continuación:



Esto conlleva a que el “technical debt” se haya reducido de 52 a 2 minutos.

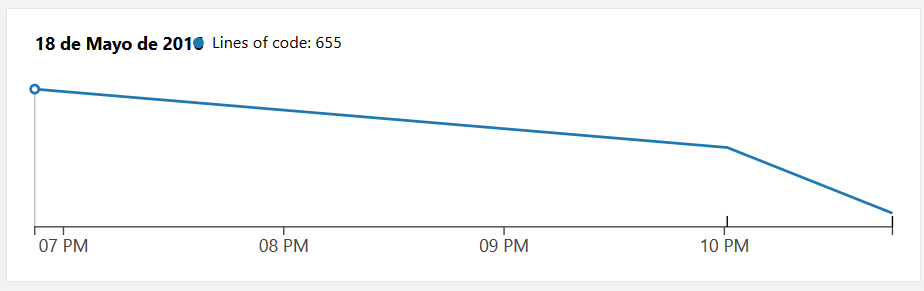
Para la parte del backend:





Se puede ver que esta parte del código es la más crítica. Es la que tiene mayor número de líneas de código, y las medidas arrojadas son bastante malas. Los “issues son 128”, de los cuales la mayoría son de tipo mayor, crítico o aún peor, “blocker”.

Se intentaron resolver los problemas reduciendo las líneas de código y quitando algunos “throw Exception”, sin embargo, a pesar de que la complejidad, las líneas de código y el “technical debt” disminuyeron, se perdían Excepciones que era obligatorio botar en caso de fallo. Debido a esto y analizando los resultados arrojados se vio que todos los “issues” que eran de tipo mayor o peor, salían debido a la cantidad de excepciones que el programa tenía que lanzar. Debido a que todas las excepciones eran fundamentales para el correcto funcionamiento del programa se decidió mantener la forma del código, respetando la forma en la que se lanzaban las excepciones. No se decidió factorizar más el código, ya que a pesar de que los métodos dentro de algunas clases son muy parecidos y contienen líneas de código idénticas, la estructura en la que el Master trabaja cada transacción es muy sistemática y clara, por lo tanto al modificar la forma en la que cada transacción se presenta se debe incurrir en muchos cambios de código y en una reestructuración de la clase y sus métodos.

Es por esto que solo se modificaron algunos “imports” innecesarios. Además, ya que la arquitectura del proyecto cambió al usar una base de datos ORACLE, se eliminaron muchas clases e interfaces innecesarias que se seguían conservando de la arquitectura anterior. Estos cambios no representaron mayor transformación en las medidas presentadas por SONAR excepto, claramente, en las líneas de código: 

Teniendo en cuenta los resultados arrojados por sonar y la forma en la que se desarrolló el proyecto se definieron las siguientes tácticas y lineamientos para mantener una aplicación modificable:

* Debido a que se usa una base de datos ORACLE la arquitectura de la aplicación cambia.
* El backend se compone de tres partes principalmente: una clase Maestro, las clases que contienen los objetos que son manejados en la lógica de la aplicación y las clases DAO.
* La clase Master se encarga de la conexión a la base de datos y el manejo de las transacciones.
* Las clases DAO se crean por cada tabla en la base de datos que necesite ser consultada (lo que implica que son pocas clases de este tipo o por lo menos son máximo la misma cantidad de tablas de la base de datos). Estas clases básicamente contienen y ejecutan una sentencia SQL, y se encargan de traer los resultados.
* El último tipo de clases son los objetos como tal que usa la lógica para ser manejados por la aplicación. Estos objetos corresponden a las entidades del modelo de negocio.
* A menos de que el modelo de entidades del negocio cambie, la cantidad de clases no va a cambiar.
* Si se quiere crear una nueva funcionalidad solo hay que seguir dos simples pasos:
  + Crear el nuevo método en la clase master. Acá se crea la nueva transacción que llama a uno o varios métodos de las clases DAO.
  + Se crean los métodos necesarios en la clase DAO, los cuales simplemente tienen las sentencias SQL.
* Si se quiere agregar una nueva entidad y asociarle a esta una funcionalidad hay que hacer los mismo pasos presentados en el punto anterior solo que se añade dos pasos iniciales en los que se crea una nueva clase para manejar el objeto en la lógica y se crea una nueva clase DAO con respecto a la nueva tabla que genera esta entidad en el modelo relacional.
* En la parte de servicios, la modificabilidad del código es muy simple ya que si se quiere agregar un nuevo servicio, simplemente se hace un nuevo método que llame al Master del backend donde se encuentra la respectiva transacción.
* Dado nuestra arquitectura en el backend, para no alterar la complejidad a la hora de modificar una funcionalidad existente o de integrar una nueva entidad al negocio, no se usan interfaces.

Estos lineamientos y tácticas fueron las que se utilizaron con el fin de tener una mejor modificabilidad y hacer más mantenible nuestra aplicación dado que se incorporó una base de datos ORACLE a nuestro proyecto y este factor hizo que replantearamos gran parte de la aplicación.