Hito 3 Proyecto E-work



Nombres: Fabio Urrea Rut:18299907-5

Felipe Villalobos

índice de tablas

[Tablas 3](#_Toc120666354)

[Resumen 3](#_Toc120666355)

[Distribución de incidentes según tipo 4](#_Toc120666356)

[Distribución mensual de Tickets según tipo de incidente reportado 5](#_Toc120666357)

[Tipo de incidente: operaciones de ciberseguridad de CSIRT de gobierno 6](#_Toc120666358)

[Tipo de incidente de código malicioso 6](#_Toc120666359)

[Tipo de incidente: seguridad de contenido de la informacion 6](#_Toc120666360)

[Tipo incidente de fraude 7](#_Toc120666361)

[Tipo de incidente: recopilación de información 7](#_Toc120666362)

[Tipo de incidente: Intrusión 8](#_Toc120666363)

[Tipo de incidente disponibilidad 8](#_Toc120666364)

[Tipo de incidente: Intentos de intrusión 9](#_Toc120666365)

[Tipo de incidente: contenido abusivo 10](#_Toc120666366)

[Tipo de incidente: otros. 10](#_Toc120666367)

[Tickets emitidos a instituciones públicas y privados 11](#_Toc120666368)

índice de ilustraciones

[Resumen anual ticket y tipos de incidentes 4](#_Toc120699582)

[Distribución de incidentes según tipo 5](#_Toc120699583)

[Distribución mensual de tickets según tipo de incidente reportado 5](#_Toc120699584)

[Tipo de incidente: operaciones de ciberseguridad de CSIRT de gobierno 6](#_Toc120699585)

[Tipo de incidente de código malicioso 7](#_Toc120699586)

[Tipo de incidente: seguridad de contenido de la información 7](#_Toc120699587)

[Tipo incidente de fraude 8](#_Toc120699588)

[Tipo de incidente: recopilación de información 8](#_Toc120699589)

[Tipo de incidente: intrusión 9](#_Toc120699590)

[Tipo de incidente disponibilidad 10](#_Toc120699591)

[Tipo de incidente: intentos de intrusión 10](#_Toc120699592)

[Tipo de incidente: contenido abusivo 11](#_Toc120699593)

[Tipo de incidente:otros. 12](#_Toc120699594)

[Tickets emitidos a instituciones públicas y privados 12](#_Toc120699595)

[Testing Driven Development 14](#_Toc120699596)

índice de contenido

[introducción 13](#_Toc120716092)

[Proyecto 13](#_Toc120716093)

[¿Quiénes son nuestros potenciales usuarios y que requieren? 13](#_Toc120716094)

[Planificación 13](#_Toc120716095)

[Visión del producto 13](#_Toc120716096)

[Visión del producto en ambiente operativo 13](#_Toc120716097)

[¿Cómo construimos este producto? 13](#_Toc120716098)

[¿Cómo aceleramos nuestro desarrollo? 20](#_Toc120716099)

[Mitigación de riesgo 21](#_Toc120716100)

[Resultado de la mitigación de riesgo 21](#_Toc120716101)

[Conclusiones 21](#_Toc120716102)

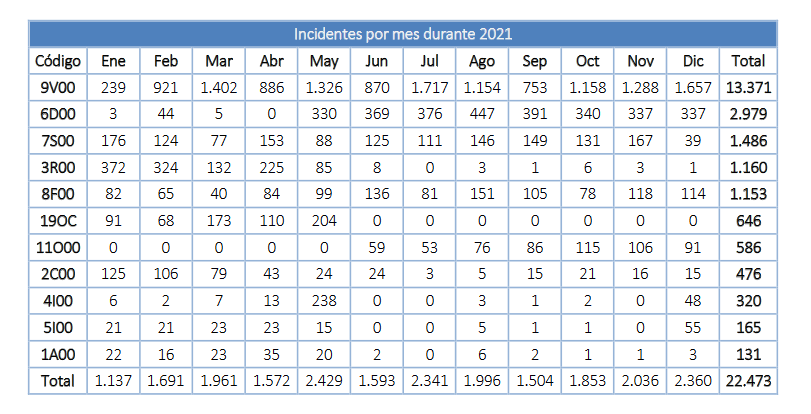
[Referencias 21](#_Toc120716103)

# Tablas

### Resumen anual de ticket y tipos de incidentes

* Podemos deducir que durante el año 2021 los mayores ataques fueron de vulnerabilidad del sistema con 13371 casos en el 2021, además la tabla nos muestra que el índice menor de casos lo muestra el contenido abusivo con 131 caos en el año. Esta tabla nos muestra que la mayor cantidad de ataques de diferentes tipos se dio en el mes de mayo con 2429 casos.

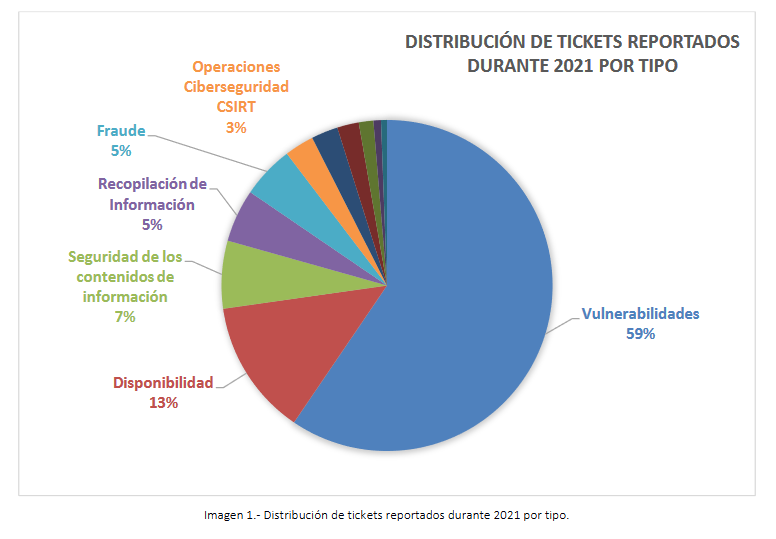




### Distribución de incidentes según tipo:

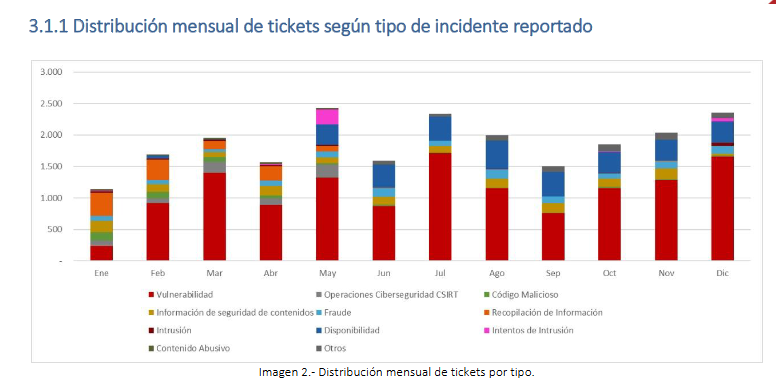
* El incidente de vulnerabilidad es el más recurrente representando un 59% del total de los ataques cibernéticos.

Debido a que las políticas de seguridad son tan deficientes están representan un 3% como operaciones de ciber seguridad (CSRIT)



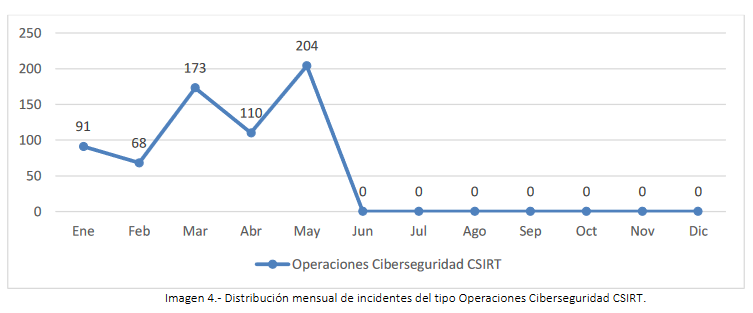
### Distribución mensual de Tickets según tipo de incidente reportado

* El mayor incidente reportado es el de vulnerabilidad con 1717 casos en septiembre y la misma cantidad de casos se produce en enero.



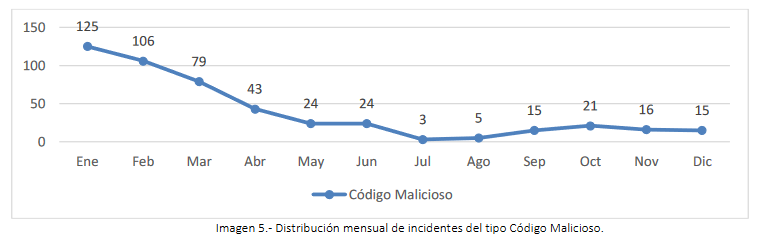
### Tipo de incidente: operaciones de ciberseguridad de CSIRT de gobierno

* En mayo se produce un peak de ataques a la red de conectividad del estado (RCE) especialmente al bloqueo de IP.



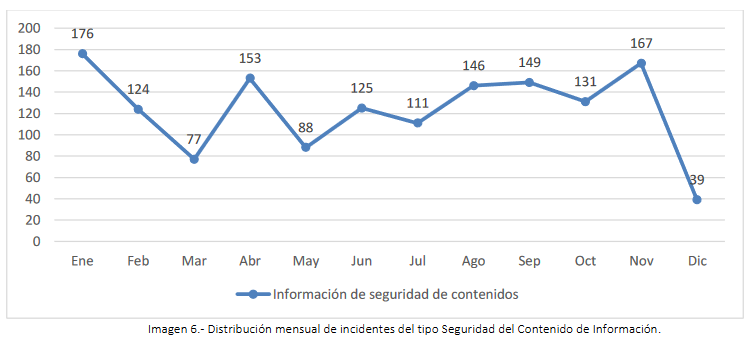
### Tipo de incidente de código malicioso:

* Este tipo de ataque conocido como software o malware que fue disminuyendo notablemente desde enero que tenia 125 casos hasta llegar a diciembre con 15 casos solamente.



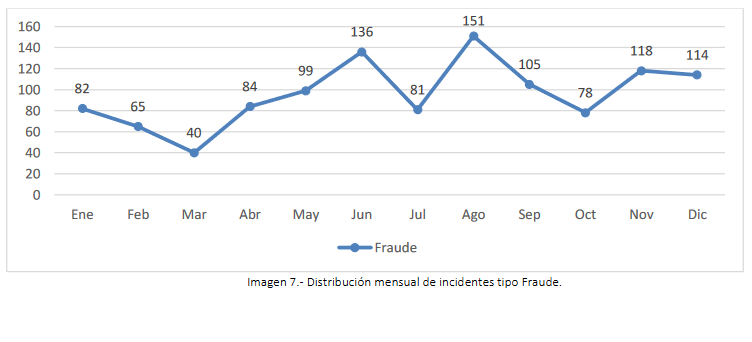
### Tipo de incidente: Seguridad de contenido de la información

* Este tipo de incidente tiene su mayor alza en enero con 176 casos, donde algún sistema fue vulnerado y modificada su información, hubo fuerte caída de este ataque en diciembre con 39 casos.



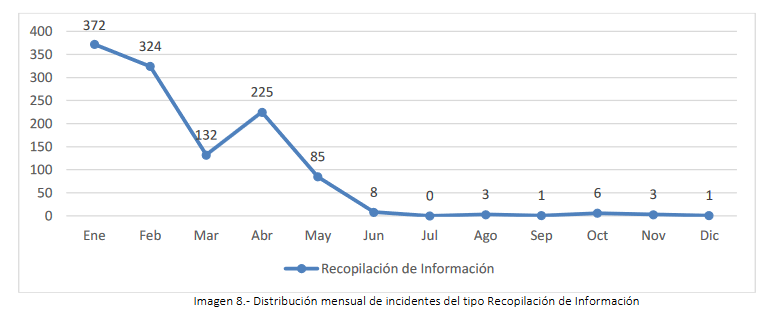
### Tipo incidente de fraude

* Este tipo de incidente conocido como phishing o spear phishing permite a los cyber delincuentes penetrar a los sistemas aprovechándose de la falta de conocimiento de los usuarios, por eso se han hecho muchas campañas para crear conciencia sobre la seguridad digital de los empleados en las compañías, la mayor cantidad de fraude se produjo en agosto llegando a 151 casos.



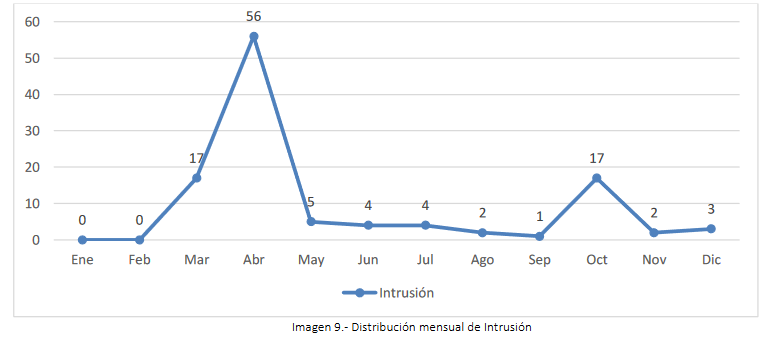
### Tipo de incidente: recopilación de información

* La recopilación de la información es el envió por parte de los delincuentes de solicitudes a un sistema para descubrir los puntos débiles de este. En enero se produjo la mayor cantidad de casos llegando a 372 y después llego a diciembre con una gran baja de 1 caso.



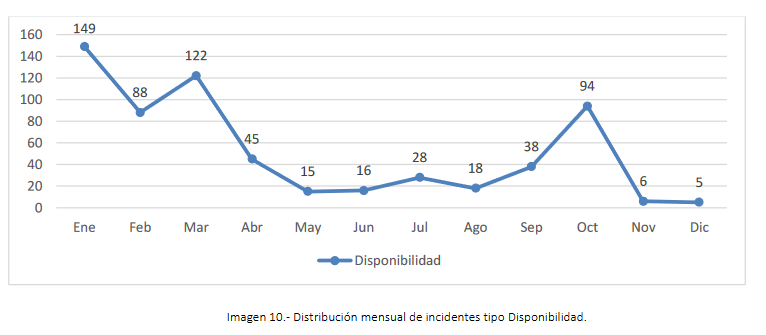
### Tipo de incidente: intrusión

* Este ataque principalmente se produce por el acceso no autorizado a través del uso de credenciales robados. La mayor cantidad de casos denunciados fue en abril con 56 casos y disminuyo a 3 casos en diciembre.



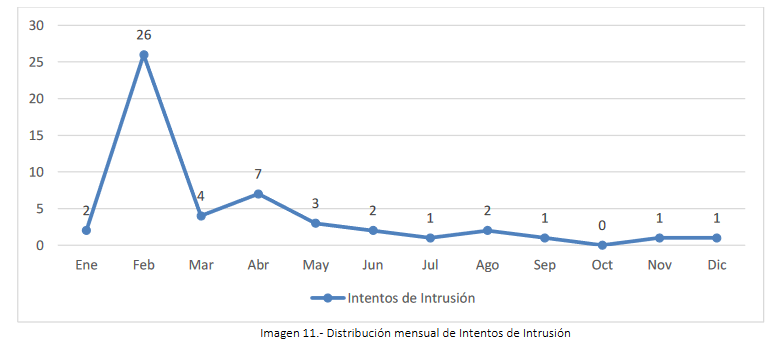
### Tipo de incidente disponibilidad

* Asociados a los ataques de denegación de servicio o Dos o DDos, este año se observo un descenso de los registros, potencialmente debido a una baja de la actividad Hacktivista contra instituciones de gobierno, lo que explicaría que los incidentes aumenten en octubre.



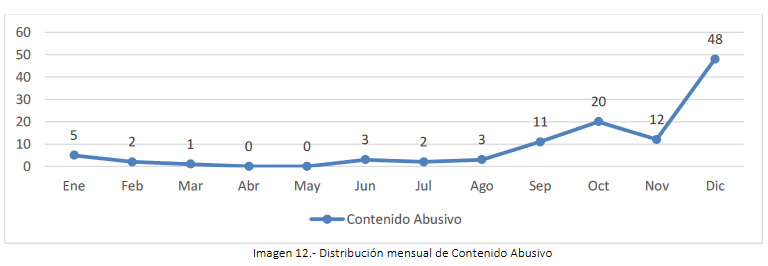
### Tipo de incidente: Intentos de intrusión

* Se trata de intentos fallidos de comprometer un sistema o interrumpir cualquier servicio explotando vulnerabilidades conocidas. En este grafico se muestra que un febrero hubo la mayor cantidad con 26 casos, pero no se tiene claridad si disminuyo notablemente un diciembre porque el CSIRT del gobierno cambio su clasificación.



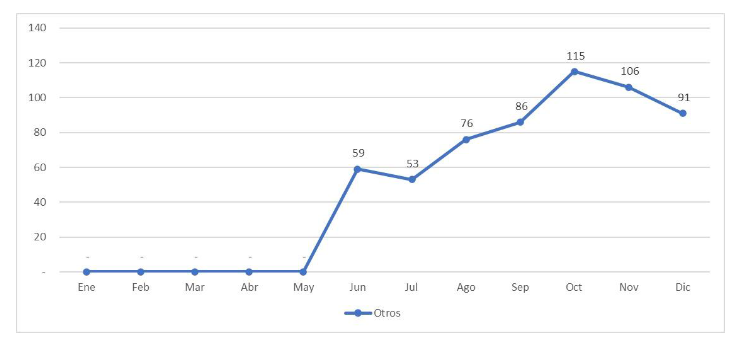
### Tipo de incidente: contenido abusivo

* En el caso de contenidos abusivos vemos que el grafico muestra un alza de casos llegando a aumentar a 48 casos en diciembre, este aumento se debe a que muchas instituciones no están tomando las debidas precauciones para evitar los ataques de agentes maliciosos que alteran las paginas web de sus victimas y redirigen el trafico hacia contenido abusivo como pornografía, violencia u otros.



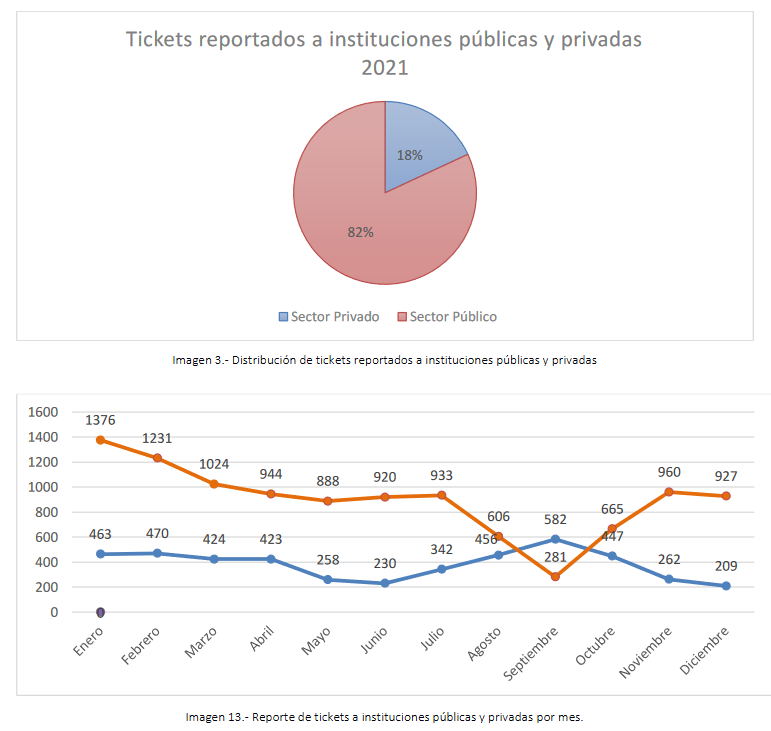
### Tipo de incidente: otros.

* Estos están clasificados como otros, ya que no entran en la categoría de clasificación dado por Enisa (agencia de la unión europea para la ciber seguridad), estos no son resultados para analizar potenciales amenazas. El mayor numero de casos de estos se produjo en octubre con 115 casos y fue disminuyendo hasta llegar a diciembre con 91 casos.



### Tickets emitidos a instituciones públicas y privados

* Desde la creación del CSIRT del gobierno este se fue vinculando con el sector privado y al unir fuerzas han logrado mantener el cyber espacio más seguro y así proteger la información de todos los chilenos. Es así como de un universo del 100% de tickets el 18% (4061 casos) hayan sido aportados por el sector privado.



# Introducción

Cuéntenos acerca de su producto – las principales características – el segmento del mercado – la competencia – cuando saldrá a mercado.

Las empresas enfrentan un cambio con los trabajadores. Las generaciones más jóvenes esperan cosas distintas de los empleos. Estamos entrando a una nueva era de autodeterminación laboral. Las empresas deberán centrarse en adaptar el trabajo a la vida de las personas y sus necesidades. Por esta razones se crea E-work como una forma de acercar o generar trabajos con características especiales para este nuevo grupo emergente de trabajadores. A pesar de ser numerosas las empresas o aplicaciones web como competencia, en E-work queremos especializarnos en trabajos por empleados independientes, trabajos sin grado académicos y trabajos en pymes para integrar a la comunidad.

# Proyecto

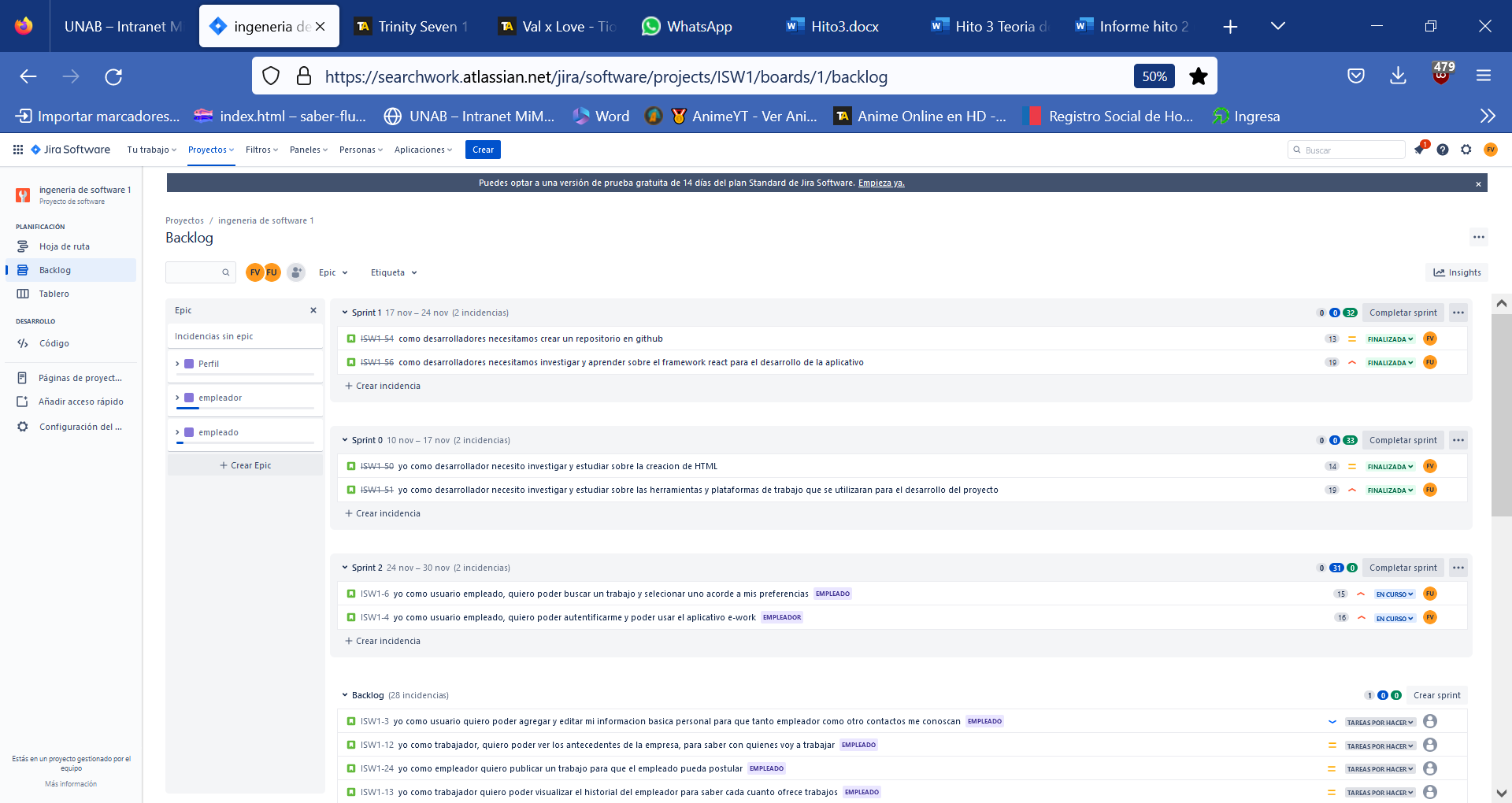
## ¿Quiénes son nuestros potenciales usuarios y que requieren?

Describir nuestros usuarios – indicar sus viajes de usuario, identificar y concluir sus necesidades como problemas y describa los features que debe tener el proyecto, por tanto, listando para ello los requerimientos y los diferentes escenarios que soportará el producto (ya sea mediante historias de usuarios o diagrama de casos de uso). Use imágenes y esquemas monísticos.

Nuestros potenciales usuarios son trabajadores que quieran usar la aplicación como método para publicitarse y expandir su alcance para poder laborar, pymes que tengan dificultad para encontrar empleados estables, empleados que busquen trabajos con flexibilidad horaria y se adecuen a sus necesidades. La particularidad de E-work es poder ofrecer a los usuarios la capacidad de elegir sus horarios de trabajo o encontrar uno que se adecue a las necesidades y preferencias. Además, promover trabajos que no requieran de un grado académico para poder inscribirse.

## Planificación

La metodología de trabajo que se utilizó en el proyecto es scrum debido a su adaptabilidad, flexibilidad, orden y organización al desarrollar un proyecto.

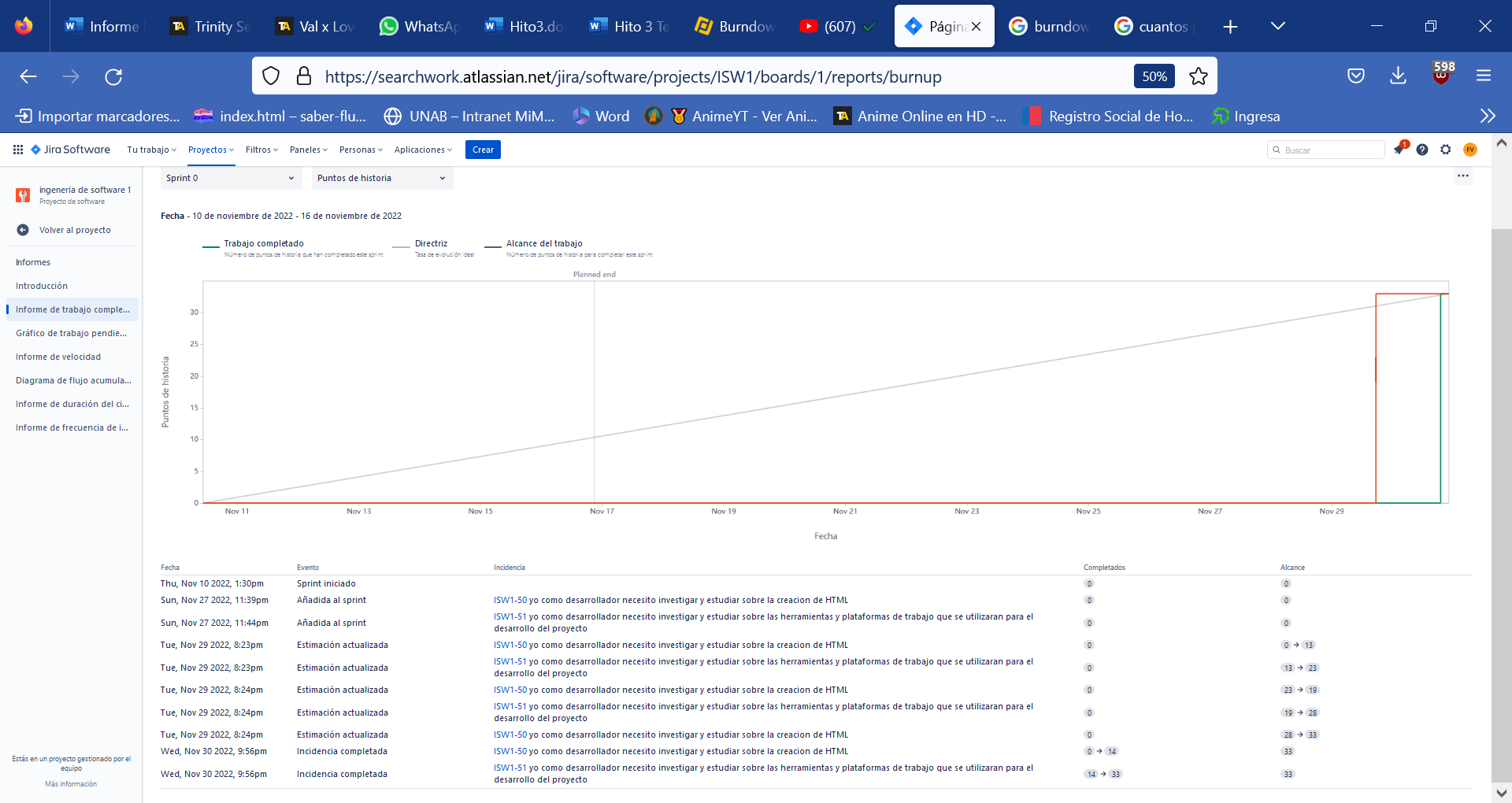


El proyecto tiene 3 sprint de 1 semana, comenzando el día 10/11/22 hasta el dia 1 / 12 /22. Se trabajará 5 días a la semana, dos horas diarias, completando un total de 10 horas a la semana y 30 horas de primera entrega de avance proyecto.

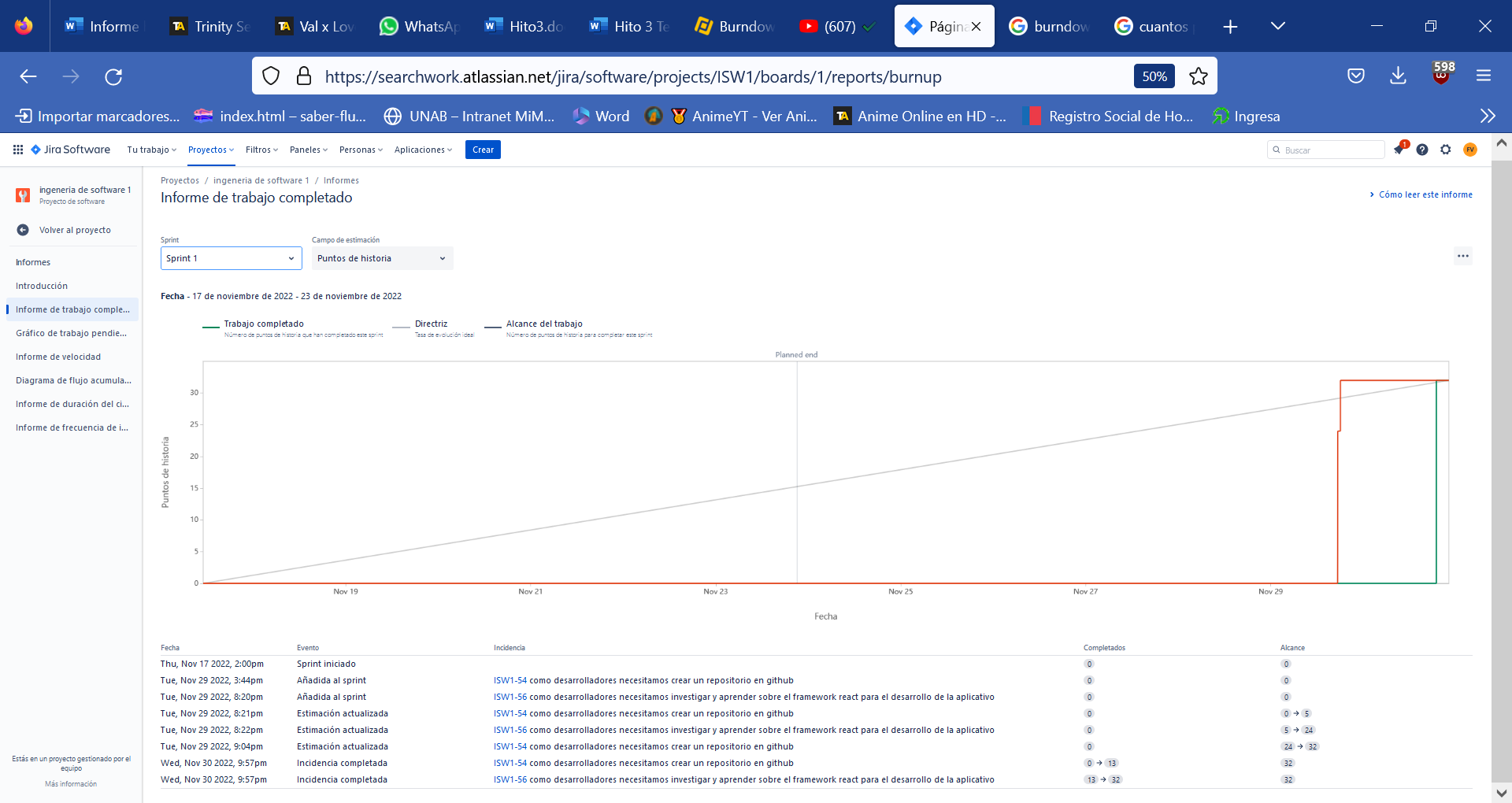


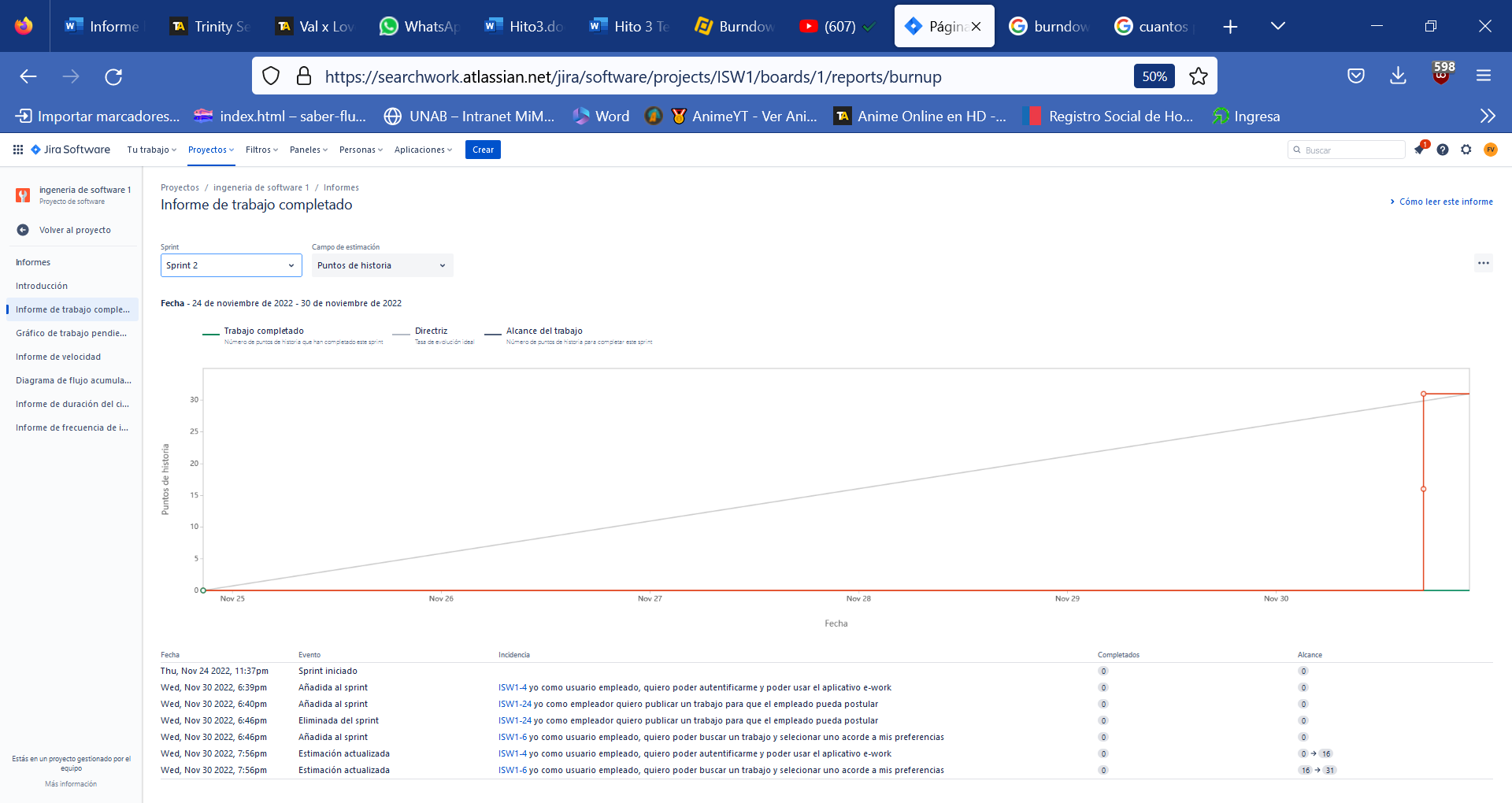
Estimamos que el proyecto podría tardar hasta 1 año en desarrollar una demo completa con todas sus features.

Sprint 0



Sprint 1





Se puede apreciar que como equipo seleccionamos las historias de usuario contenidas en cada sprint para crear el primer demo del proyecto y las dividimos en tareas que cumplimos durante el desarrollo del sprint. Los gráficos de burndown chart indican que como grupo no cumplimos con la metodología de trabajo ya que no hicimos un buen uso de la herramienta jira, pero como equipo desarrollador cumplimos con la fecha de entrega de la demo.

Cuál es el roadmap del producto – en cuantos años veremos todos estos features (use el roadmap de jira) – cual es la metodología de gestión y la de desarrollo (sugerencia scrum + cascada) + muestre el producto backlog priorizado y estimado + hable de cuantos puntos de historia tiene + cuanto durarán los sprints (4 semanas) y cuáles son las historias de usuario – como se subdividen las tareas en tareas y como funciona y lo apoya jira en esto – considere el burndown chart.

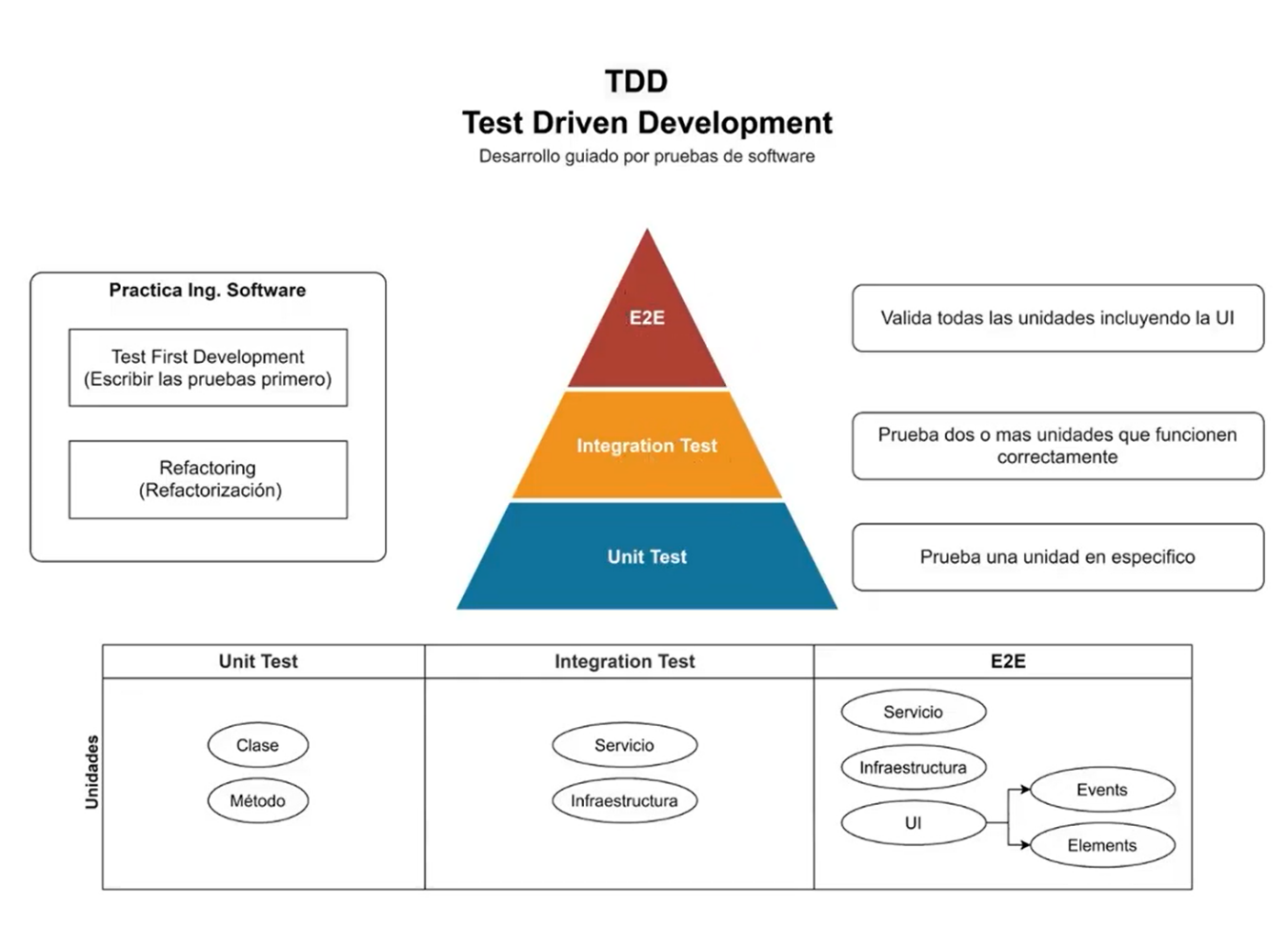
|  |  |
| --- | --- |
| **Planificación – se evalúa** | Las tareas son trazables a historias de usuario/requisitos |
|  | Las tareas tienen estimación de esfuerzo/complejidad |
|  |  |
|  |  |

# Visión del producto

## Visión del producto en ambiente operativo

## ¿Cómo construimos este producto?

Empleando la metodología DevOps usada en el desarrollo de software se hará uso de diferentes herramientas con la finalidad de automatizar procesos de testing.

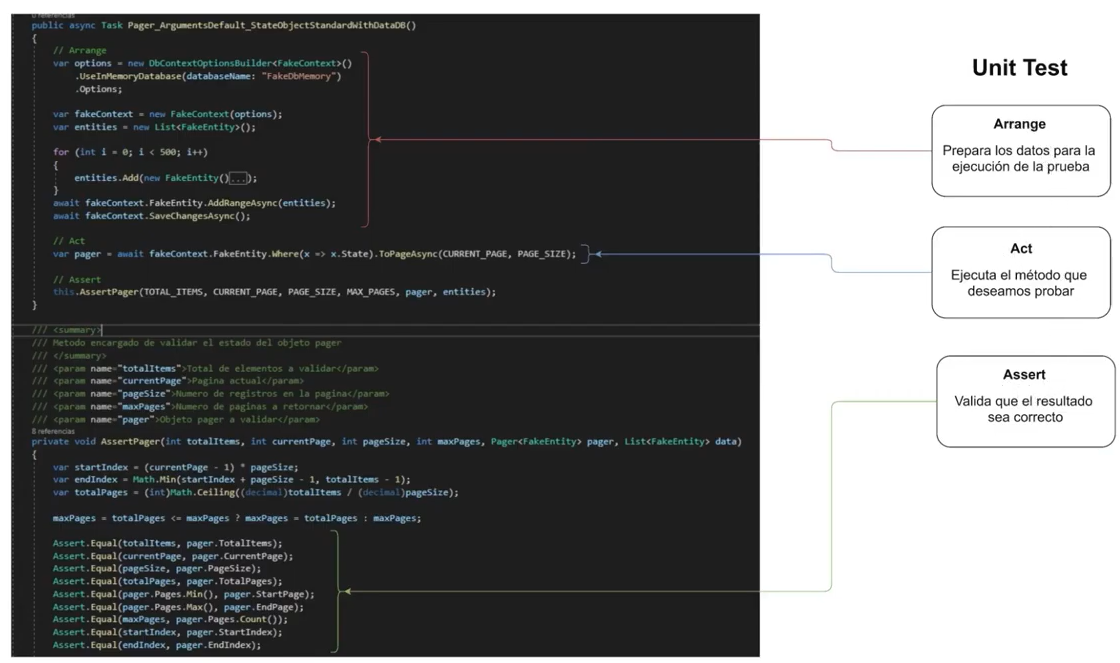
 Los testeos que se buscara automatizar corresponden a pruebas unitarias, pruebas de integración y pruebas de aceptación.

* Las pruebas unitarias: testearan lo que corresponde a las unidades que contienen, ya sea un método o una clase del programa, en este caso se va a verificar que lo que codificamos se haga y se haga bien.
* Las pruebas de integración: tienen la capacidad de probar dos o más unidades que funcionan correctamente, así como infraestructuras y servicios como lo son por ejemplo una clase con integración donde se debe verificar que la comunicación entre dos puntos sea
* Las pruebas de aceptación: pruebas que verifican de manera formal la conformidad del sistema con los requisitos y la puesta en producción. Esta prueba mide si se cumplen los requisitos contra actuales que pidió el cliente.

Las Pruebas tanto de integración como las pruebas unitarias es posible automatizarlas usando los framework “XUnit” y “.Net Core”.

Se empleará la metodología DevOps para mantener la comunicación entre la parte de Developers y la parte de operaciones, la principal característica del operador es mantener el código de los developers en funcionamiento y la de los developers es crear guías para que este entienda que podría salir mal, como se puede arreglar, que saldrá bien y como se puede observar, en el fondo cómo funciona el servicio. Con DevOps se obtendrán menos errores, los deployment serán más seguros, más rápidos y más confiables. Se pueden automatizar diferentes tipos de prácticas entre ellas los test que se están aplicando en este informe. Es posible medir la productividad como por ejemplo cuanto tarda un fix en salir o cuanto se tarda un operador en arreglar un error y por último esta metodología permite mejorar las herramientas internas compartidas por el o los equipos.

Como bien se había dado una pequeña descripción de lo que era una prueba unitaria, también es necesario decir que su estructura esta compuesta por tres partes las cuales son:



Arrange (Preparar): En esta etapa se preparan los datos necesarios para la prueba unitaria

Act (Ejecutar): Ejecuta el método que vamos a probar

Assert(Validar): En el assert validamos que la prueba de este método sea el esperado según lo que necesitamos.

## Docker

* Si bien para la etapa que corresponde a Arrange y Act es recomendado usar los frameworks como XUnit y .Net Corepara su automatización, la parte de Assert es posible automatizarla usando SonarQube con SonarScanner.
* Para el uso de estas dos últimas herramientas se usará Docker que automatiza el uso de varias aplicaciones generando un gasto menor de los recursos disponibles en una máquina, mediante la creación de contenedores que a diferencia de una máquina virtual en la que se debe designar un espacio estático, los contenedores usan los mismo recursos y espacio por lo que cada instalación en más liviana que instalar un sistema operativo nuevo por cada aplicación que se necesite usar.

## Instalaciones de contenedores y su propósito

* Los contenedores son creados con Docker corriendo en tiempo real y con la utilización de comandos por CMD

## Contenedor de SonarQube:

* herramienta que ayuda a hacer análisis estáticos y sirve para mejorar la calidad del código.

|  |
| --- |
| Por medio del siguiente comando se crea el contener de SonarQube: |
| docker run -d --name sonarqube -p 9000:9000 -p 9092:9092 sonarqube |

* Para desplegar la interfaz de usuario se usa localhost:9000
* Al crear un nuevo proyecto en SonarQube se crea un nombre de proyecto, un token, se especifica el lenguaje en el cual está escrito el código y el sistema operativo en el que se está llevando a cabo el Scan y este generara un código para el SonarScanner.

## Métricas SonarQube:

* SonarQube nos muestra las siguientes métricas las cuales serán explicadas brevemente para su entendimiento, para información extra sobre métricas de las métricas se puede consultar en referencias del documento.

## Evidencias/Issues

Fragmentos de código de un proyecto que incumplen reglas establecidas para cada lenguaje en su respectivo plan de calidad, las evidencias se subdividen a su vez en categorías de evidencias que solo serán mencionadas:

* Nuevas evidencias
* Nuevas evidencias seguridad
* Falsos positivos
* Evidencias abiertas
* Evidencias reabiertas

## Duplicados

Ayuda a mantener una programación limpia encontrando duplicidades en el código con constantes y métodos aplicados. Las métricas entregadas por los duplicados serán mencionadas:

* Archivos duplicados
* Líneas duplicadas
* Porcentaje de líneas duplicadas (%)

## Complejidad

Se mide la dificultad del mantenimiento y escalabilidad del proyecto evaluado en SonarQube, esta métrica busca que el proyecto sea un código lo más sencillo posible.

Se mide la complejidad del proyecto por:

* Complejidad ciclomática
* Complejidad cognitiva

## Tamaño

Aquí se mide la magnitud del proyecto, no indica nada respecto a la calidad del proyecto.

## Pruebas o cobertura

Indica cuantas líneas de código son cubiertas por pruebas unitarias. Es importante destacar que SonarQube no ejecuta las pruebas ni genera informes. Solo importa informes generados previamente. Podemos encontrar información sobre los ficheros afectados, resultados sobre pruebas unitarias ya ejecutadas y medir por medio de un reporte histórico la evolución de la productividad.

## Severidades de los fallos

Se califican las severidades dentro de los siguientes puntos:

* **Vulnerabilidades**: fallas en la seguridad del programa que deben corregirse de inmediato
* **Bug**: errores encontrados en el código.
* **Security Hospot**: es un tipo de vulnerabilidad que puede o no ser corregida debido a que representa un bajo riesgo a la seguridad y queda sujeto a juicio del desarrollador si este es corregido o no.
* **Code smell**: código que dificulta el mantenimiento del proyecto.

Los siguientes términos serán usados más adelante por lo cual es importante destacar su significado.

* **Blocker**: Error con una alta probabilidad de afectar el comportamiento de la aplicación en producción.
* **Crítico**: Urge su corrección, aunque no suponga una afectación al comportamiento de la aplicación en producción o porque es una falla de seguridad de impacto alto.
* **Major**: Defecto de calidad que puede afectar enormemente a la productividad del desarrollador.
* **Minor**: Defecto de calidad que puede afectar ligeramente la productividad del desarrollador.

## Fiabilidad

capacidad de un sistema para funcionar libre de fallos dentro de sus funciones especificadas. Mide el coste que tendría arreglar esos fallos si se presentaran y para ello se entrega una escala:

* Sin Bugs
* Bug Minor
* Bug Major
* Bug Critico
* Bug Blocker

## Seguridad

Nivel de protección otorgado a los datos para que estos no sean leídos o modificados por terceros, se mide la capacidad del sistema para dar una respuesta frente ataques informáticos y reducir el número de estos, midiéndolos según la siguiente escala:

Vulnerabilidades de tipo:

* Minor
* Major
* Critico
* Blocker

## Mantenibilidad

Capacidad del producto software para ser modificado efectiva y eficientemente en respuesta a necesidades evolutivas, correctivas o perfectivas.

## Contenedor SonarScanner:

Con él se prueban algunos códigos de PHP y Python que al ser analizados muestran errores que deben corregirse a nivel de código, pero no se muestran vulnerabilidades de seguridad en los dos escaneos realizados.

|  |
| --- |
| Por medio del siguiente comando se crea el contener de SonarScanner: |
| docker run -it --rm sonarsource/sonar-scanner-cli:4.5 –version |

|  |
| --- |
| El SonarScanner se puede descargar de: |
| https://docs.sonarqube.org/latest/analysis/scan/sonarscanner/ |

Se debe descomprimir en una carpeta y en esta crear el .txt "sonar-project.properties" para la configuración del Scan que se realizara al código que se busca analizar. El .txt debe llevar el siguiente contenido:

# must be unique in a given SonarQube instance

sonar.projectKey=my:project

# --- optional properties ---

# defaults to project key

#sonar.projectName=My project

# defaults to 'not provided'

#sonar.projectVersion=1.0

# Path is relative to the sonar-project.properties file. Defaults to .

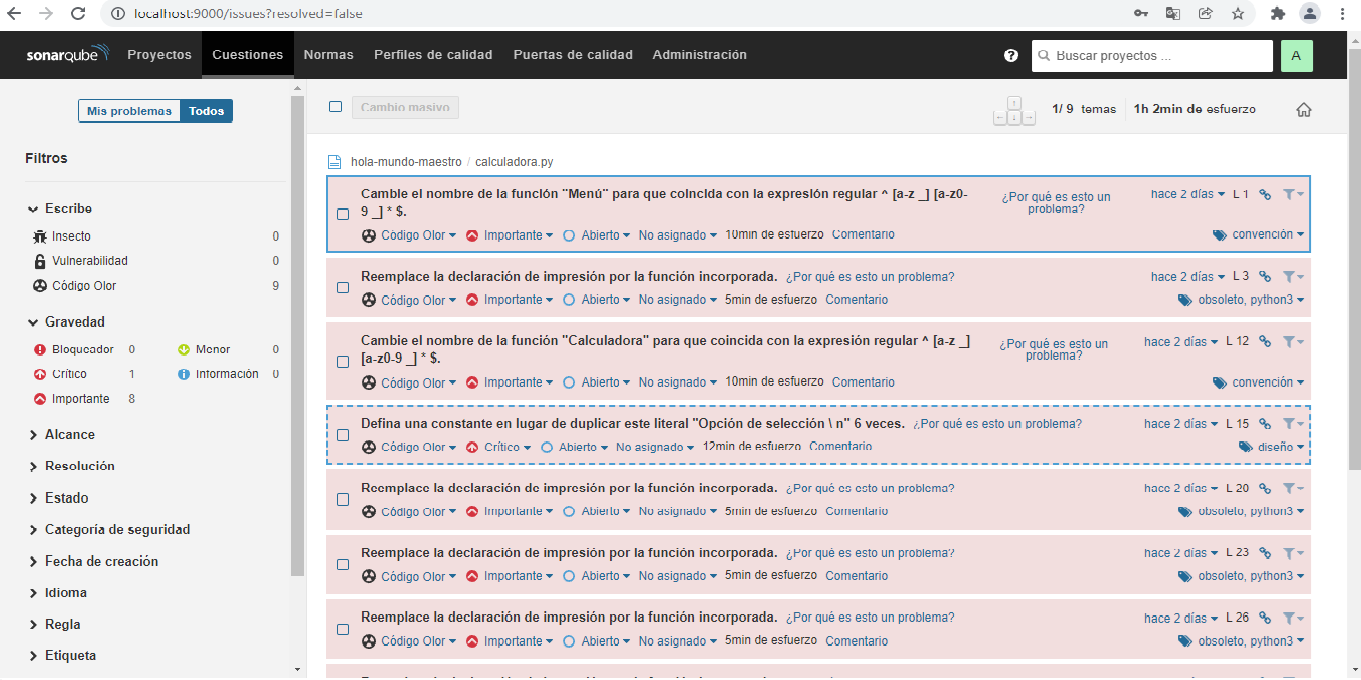
#sonar.sources=.

# Encoding of the source code. Default is default system encoding

#sonar.sourceEncoding=UTF-8

Donde sonar.projectKey debe llevar el nombre del código a analizar.

Escaneo de calculadora.py por SonarScanner, vulnerabilidades encontradas 0, errores en el código 9.



## ¿Cómo aceleramos nuestro desarrollo?

# Mitigación de riesgo

El inicio en los trabajos demoraba al principio mucho tiempo, el equipo encargado del proyecto se reunía y pensaban muchas semanas organizando las actividades antes de entregar el trabajo encomendado por el cliente.

El inicio de Thoughworks fue desarrollado principalmente por Luke Barret el 2004, el cual describió y desarrollo aún más las técnicas de inicio de un trabajo.

Esto permitió que, aunque un proyecto varia del otro generalmente se estandariza entre el negocio (no importa cuál sea su rubro) y los técnicos encargados del proyecto, lo que permite crear una lista ordenada de historias de usuario con estimación junto con un plan de lanzamiento.

The lean Inception vino a modificar lo esbelto en el 2006.

El nuevo estilo de inicio es resuelto por 2 razones:

* A) La duración del inicio es mas corta eliminando todo lo que no era sobre el producto en sí. Ej: arquitectura, proyecto, otros. Haciéndolo más delgado.
* B) El resultado final del inicio es la comprensión de MVP un concepto principal del movimiento LEAN START UP.

¿Cómo llega a realizar la MVP?

Un inicio esbelto es útil cuando el equipo necesita desarrollar o repetir varias veces un proceso como MVP.

El MVP lo construimos para saber si vale la pena continuar construyendo un producto, por lo cual se eligen funciones basadas en nuestras suposiciones, pensando en lo que es mas valioso para el usuario. Hay dos tipos de proyectos los grandes y los pequeños.

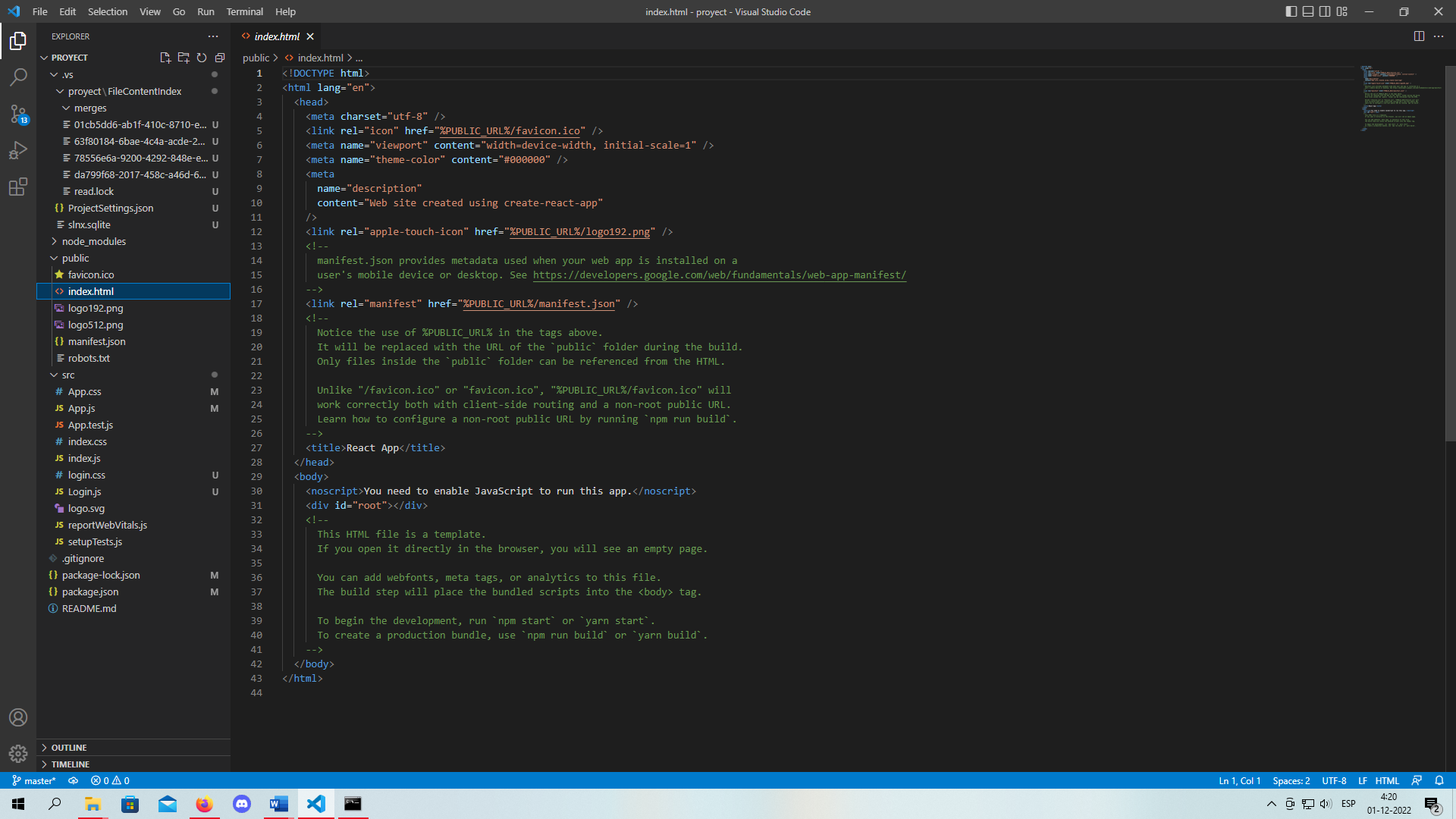
* A) grandes proyectos: encuentran que es importante un inicio esbelto para comenzar rápidamente, lo que significa que construyen iteraciones tempranas para descubrir y probar que características son realmente importantes para el usuario.

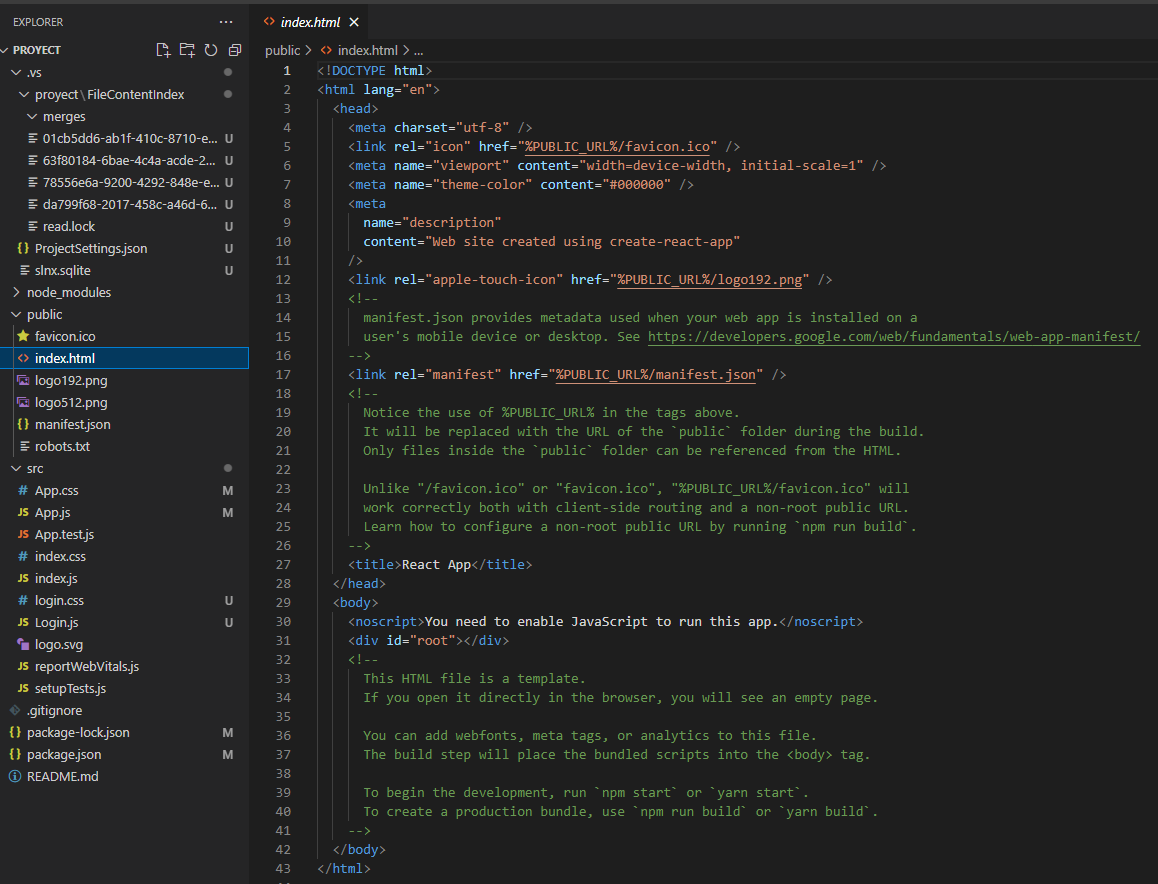
Los proyectos pequeños toman ideas de otros grandes proyectos que han sido probados por algunos MVP previamente al software y las convierten en un producto software.

El MVP, no sustituye las demás sesiones de ideación de un proyecto, la investigación del cliente, la revisión arquitectónica o el análisis competitivo. El MVP es un análisis especifico que forma parte de la comprensión que se necesita para crear un producto exitoso y así ganar tiempo para dirigir el esfuerzo en lograr el producto adecuado a las necesidades del cliente.

# Resultado de la mitigación de riesgo

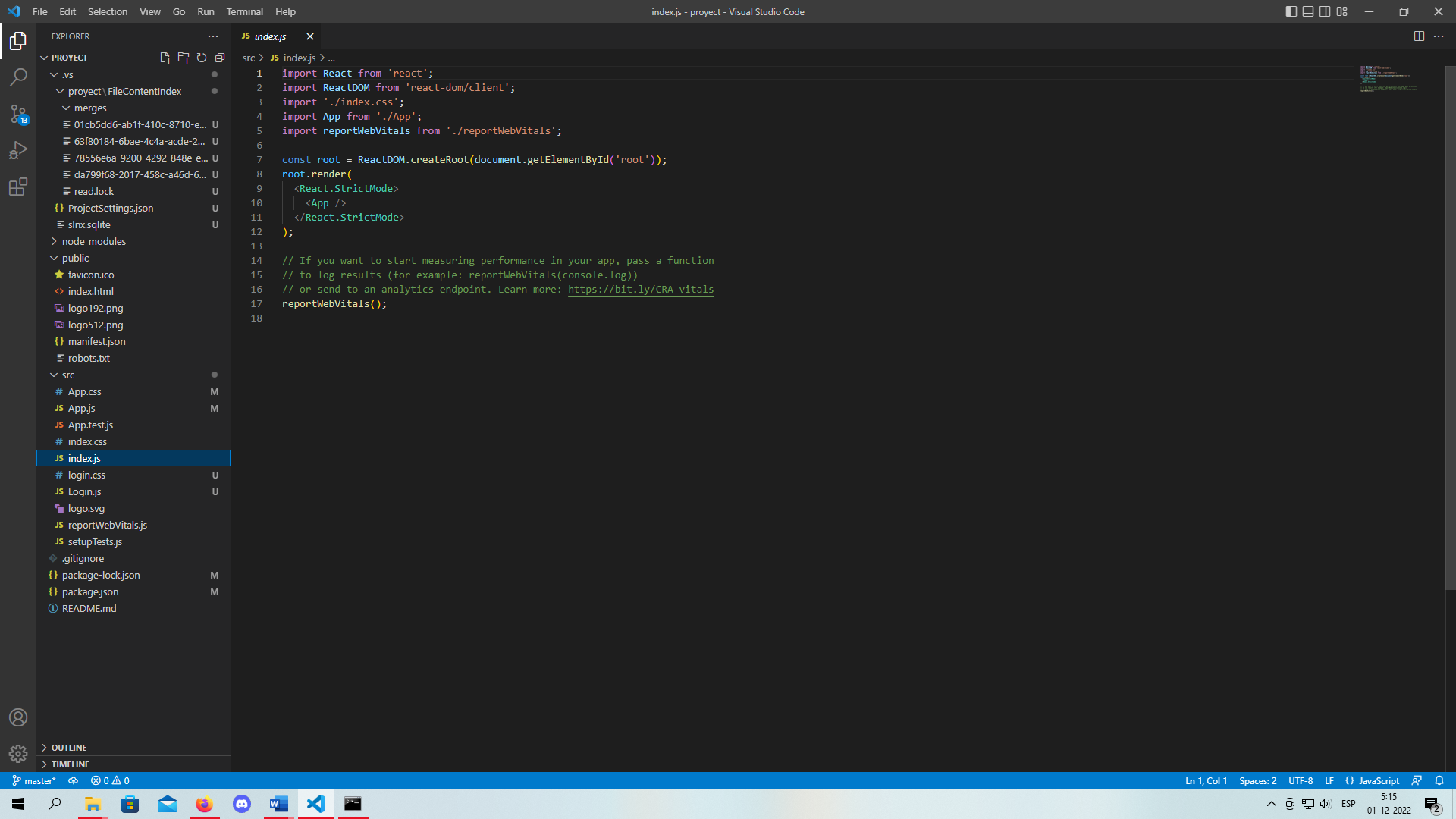
Estas dos imágenes corresponden al código dentro del archivo index.html . Este archivo está dentro de los directorios de los servidores de los sitios webs que se carga siempre que se solicita un dominio y no se especifica el nombre de un archivo específico. Es la página principal o el home de nuestra web sin ella no existe el sitio web, de ella salen las demás ramificaciones que en este caso serían las vistas que el usuario va explorando conforme va usando el sitio web de E-work.



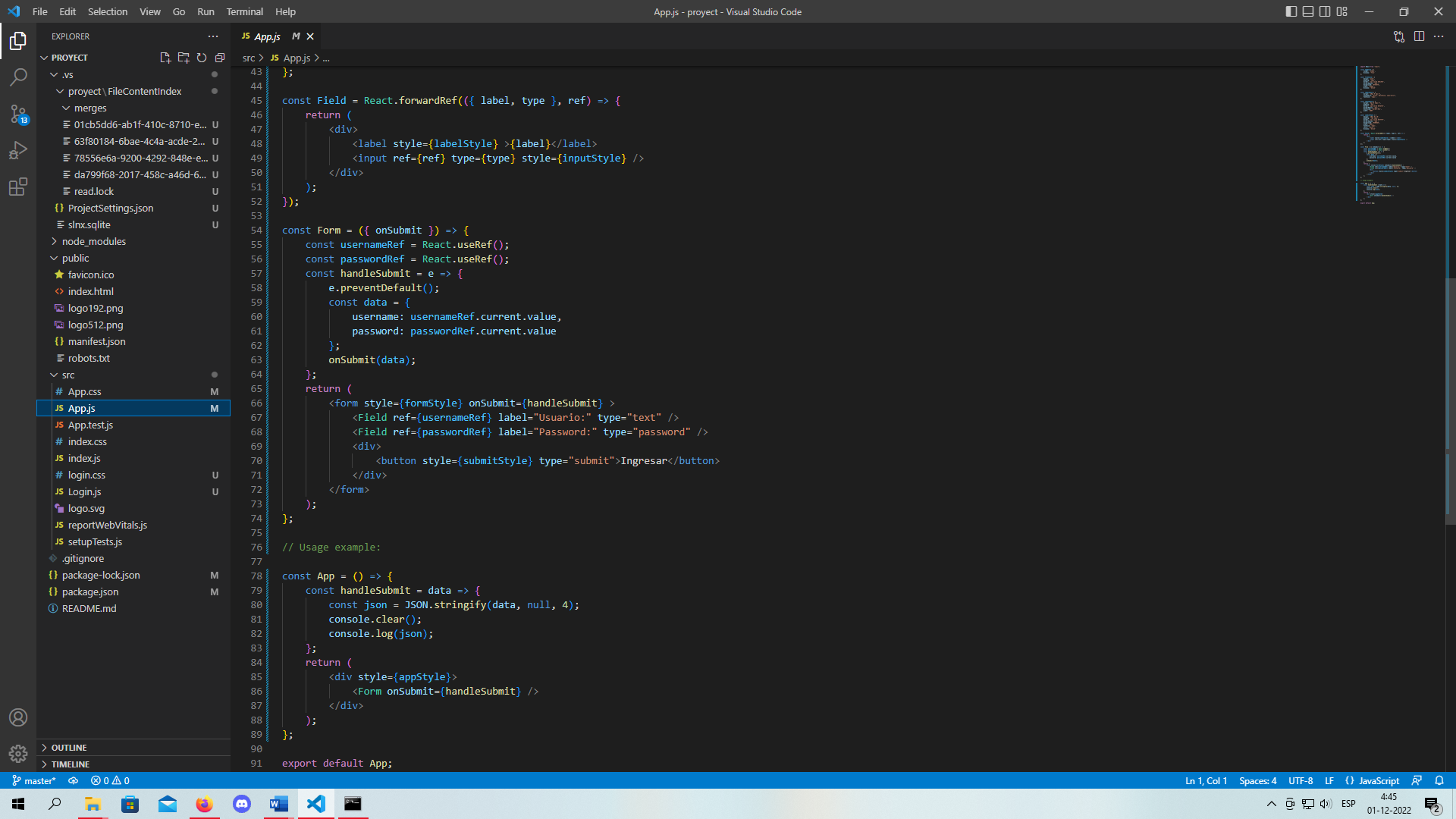


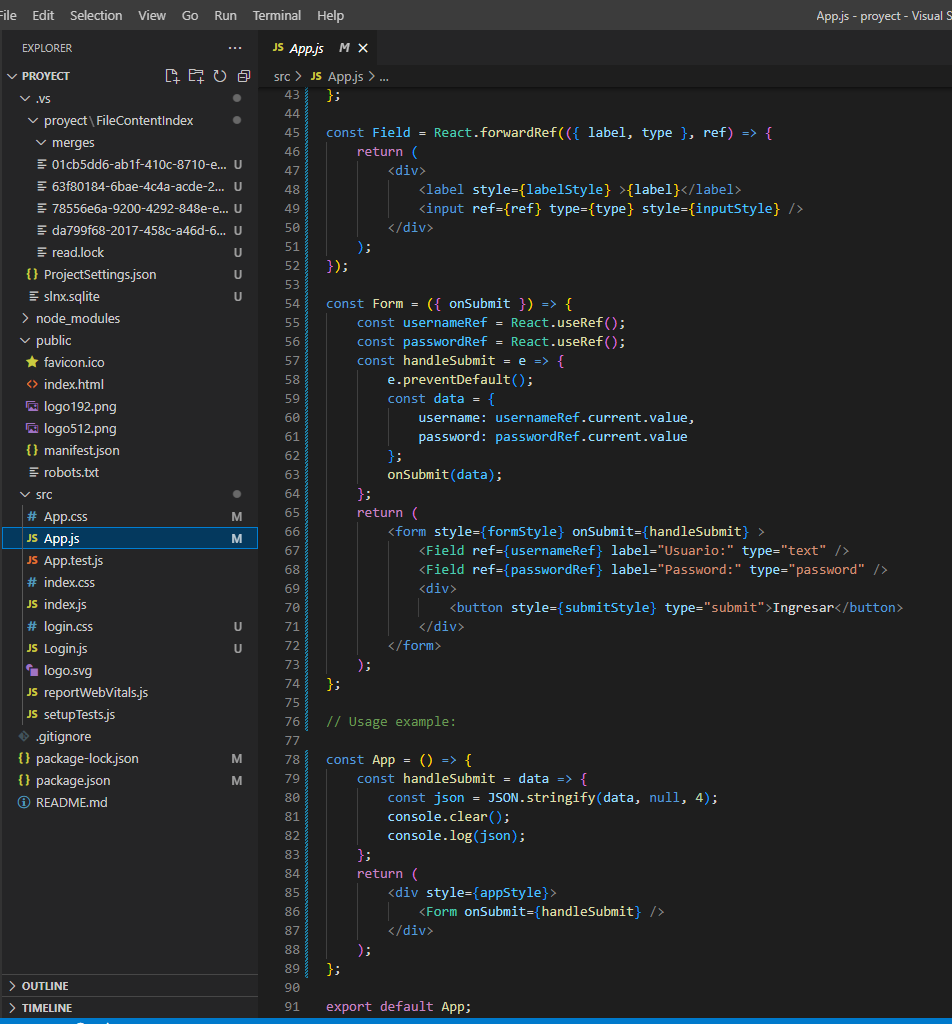
Index.js de este archivo el componente App.js está siendo utilizado. El archivo comienza importando todos los módulos JS y otros activos que necesita para ejecutarse.

Index.js

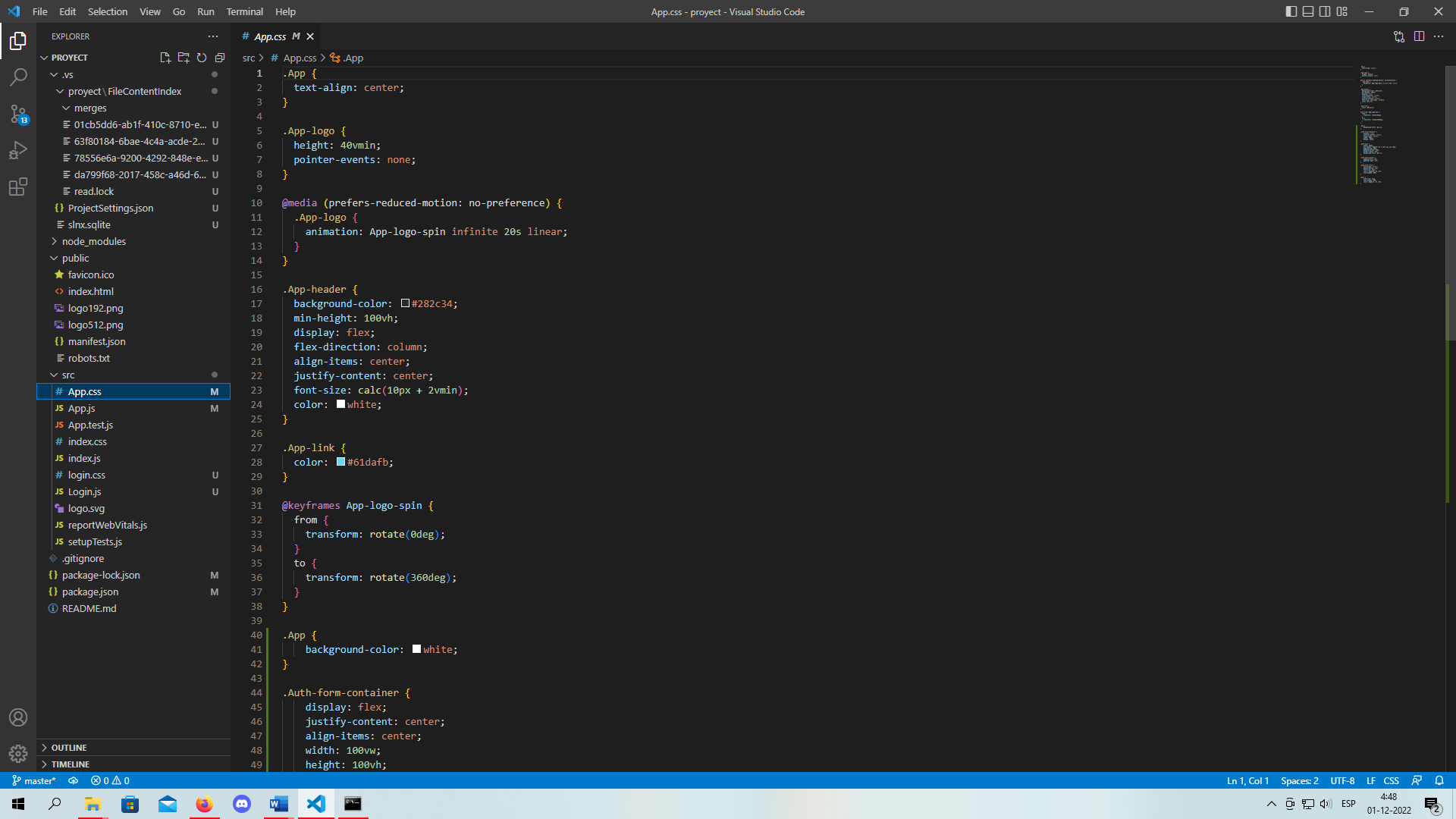


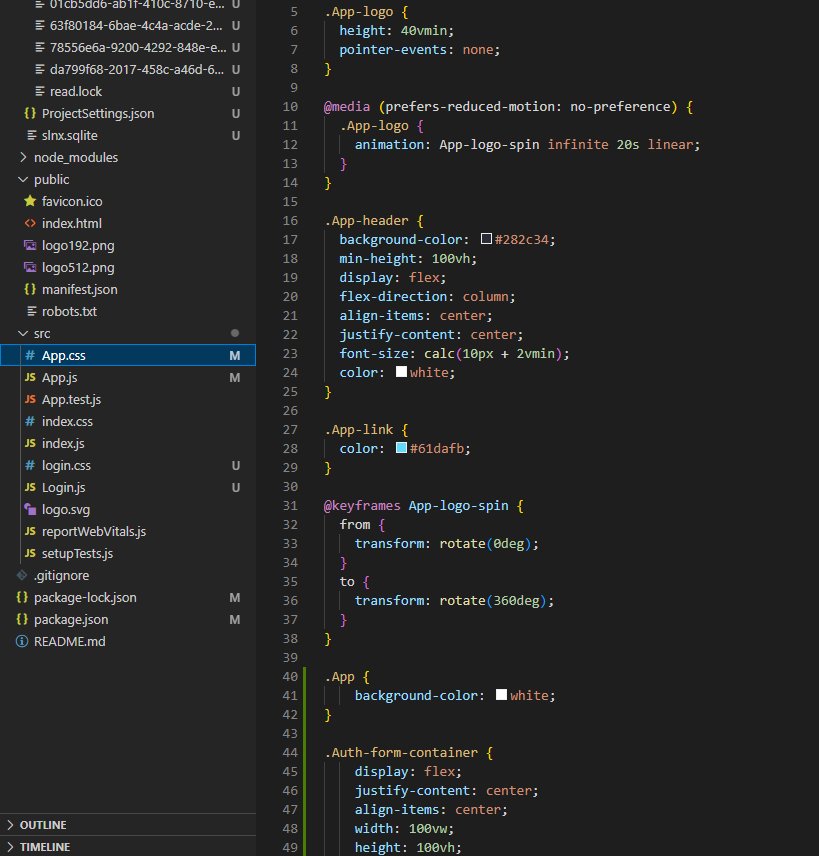
App.js



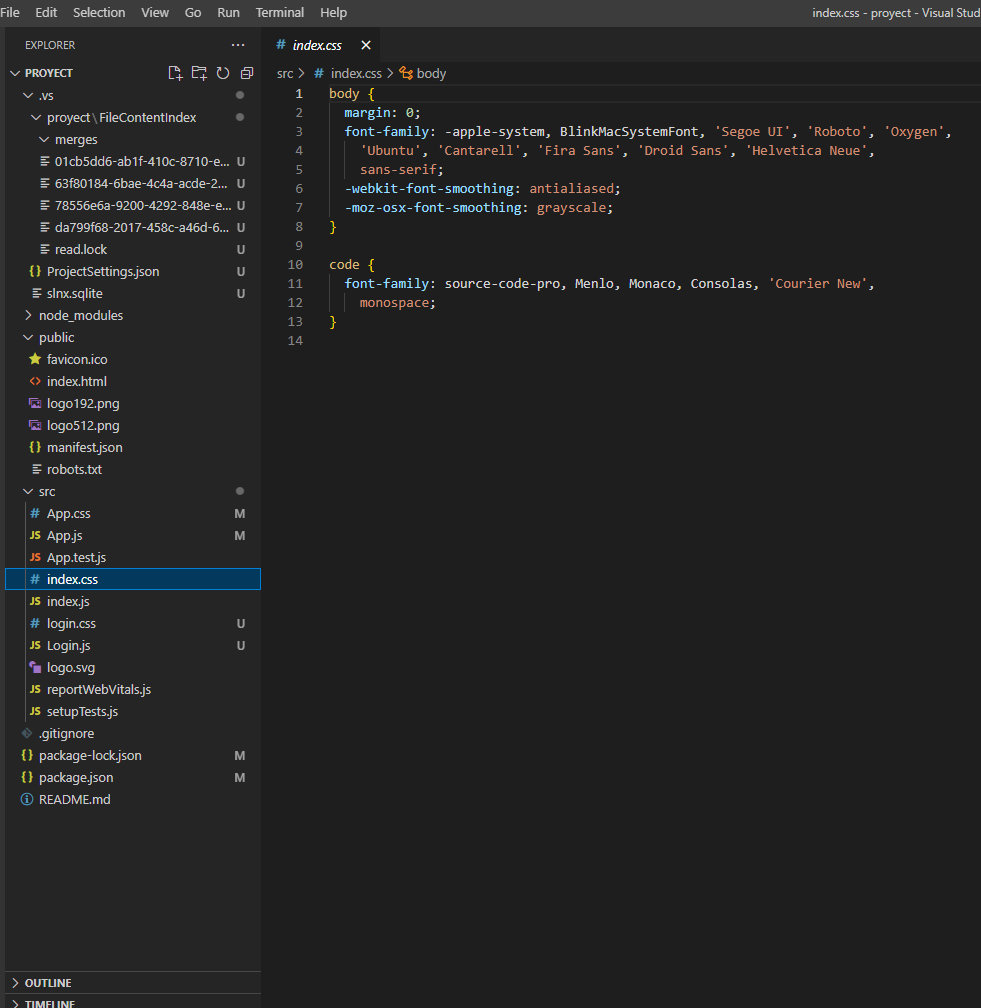


App.css contine los estilos que se le aplicaron al archivo app.js

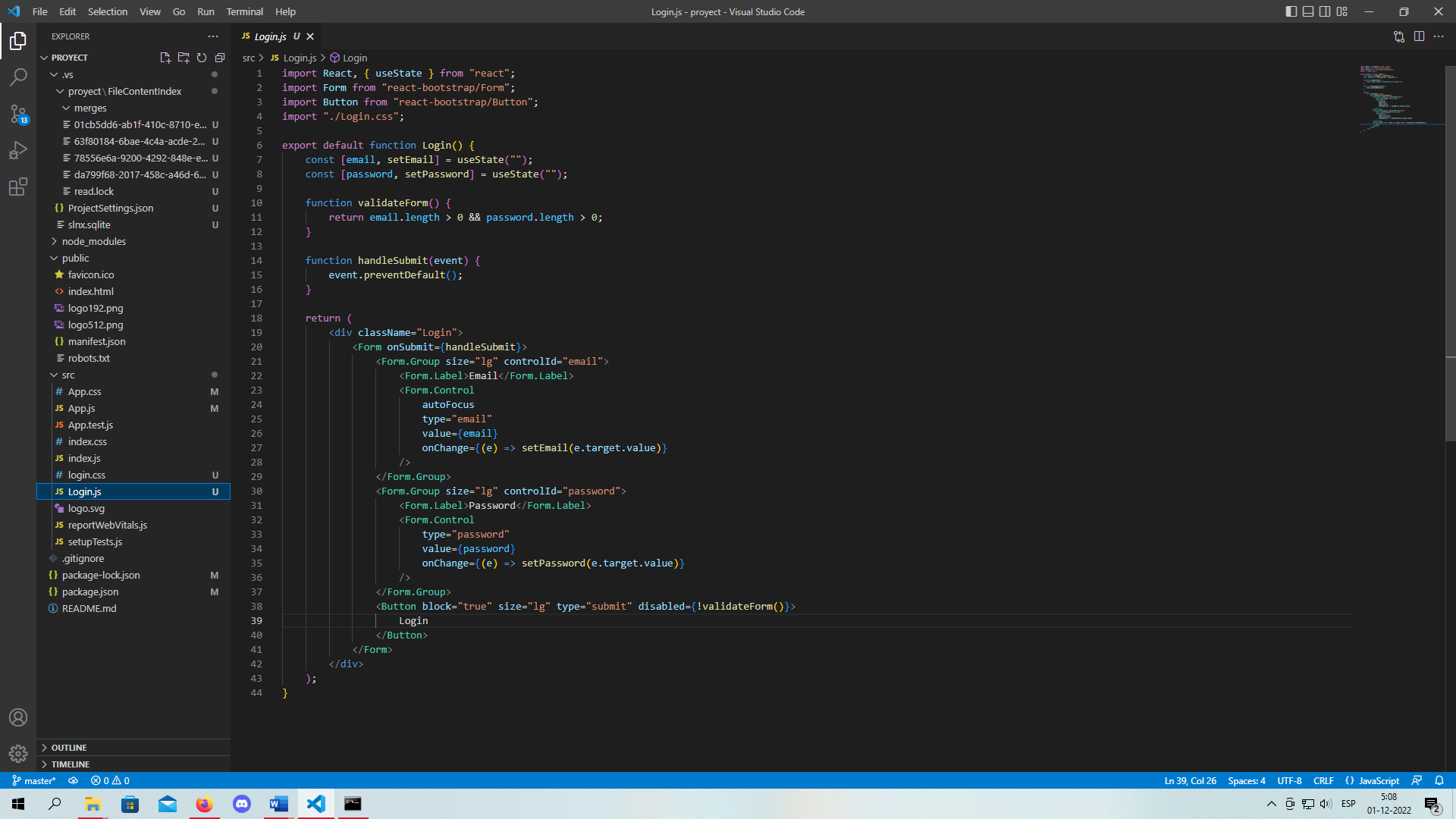


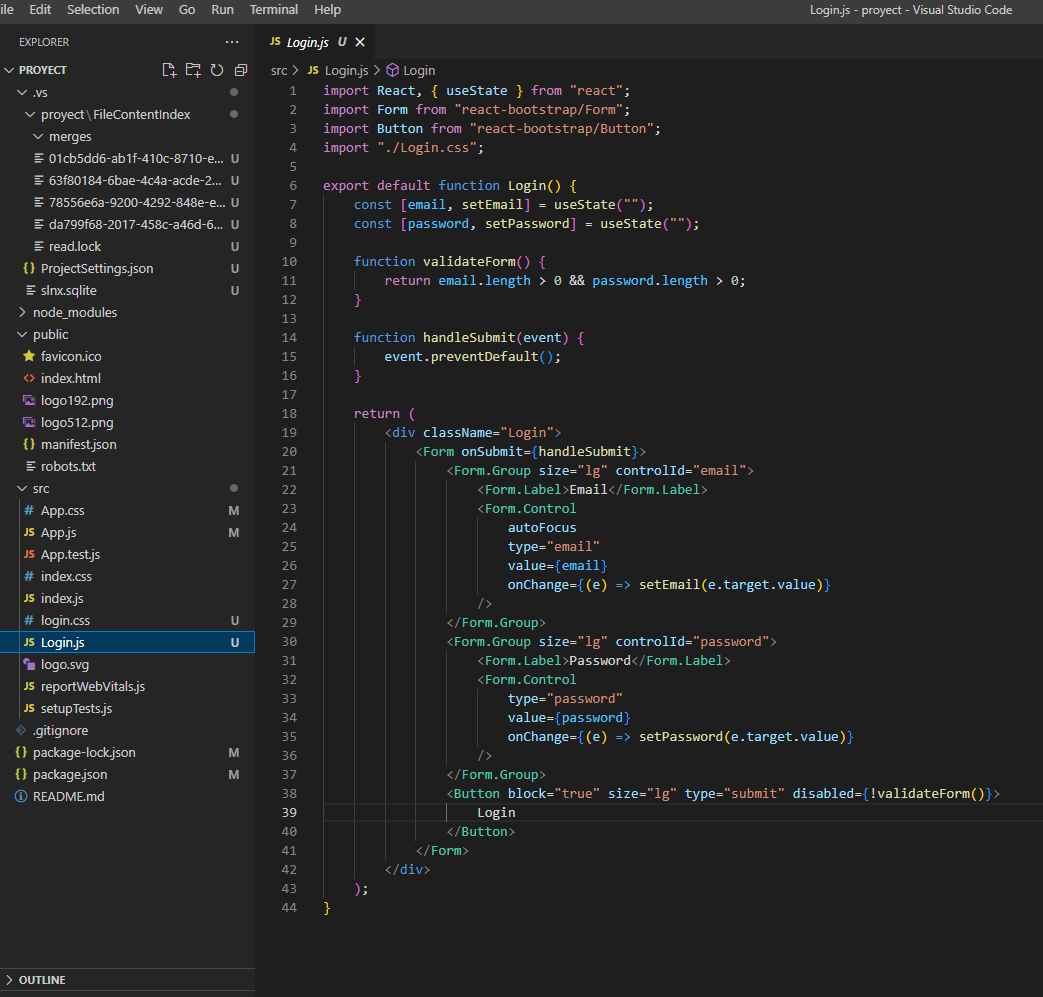


Index.css archivo que contiene el código que le brindara el estilo por defecto a la pagina

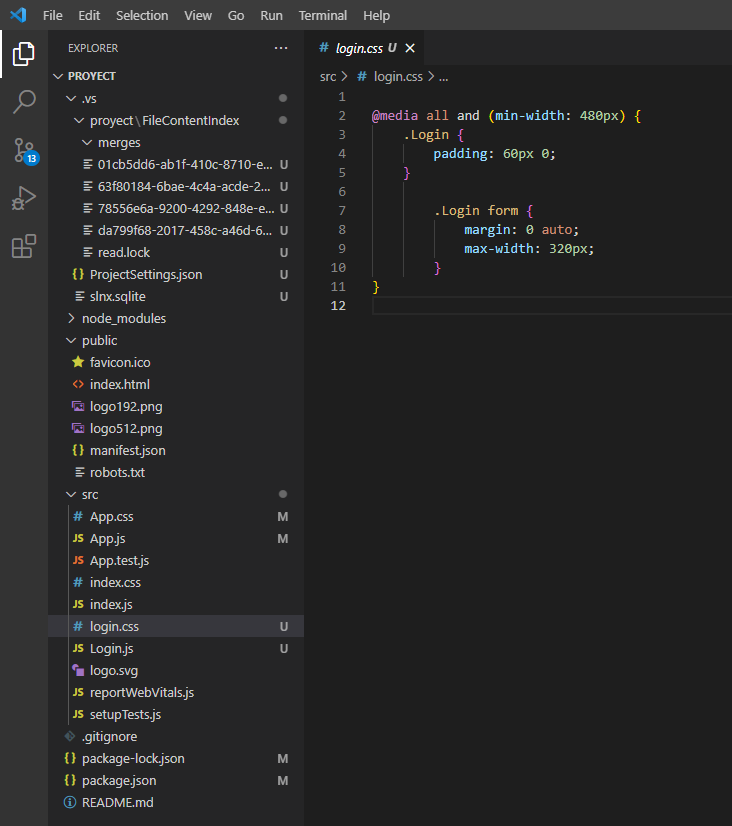


Login.js el código que se encuentra aquí recibe los parámetros que el usuario entrega al completar las casillas de input y conecta con la base de datos.

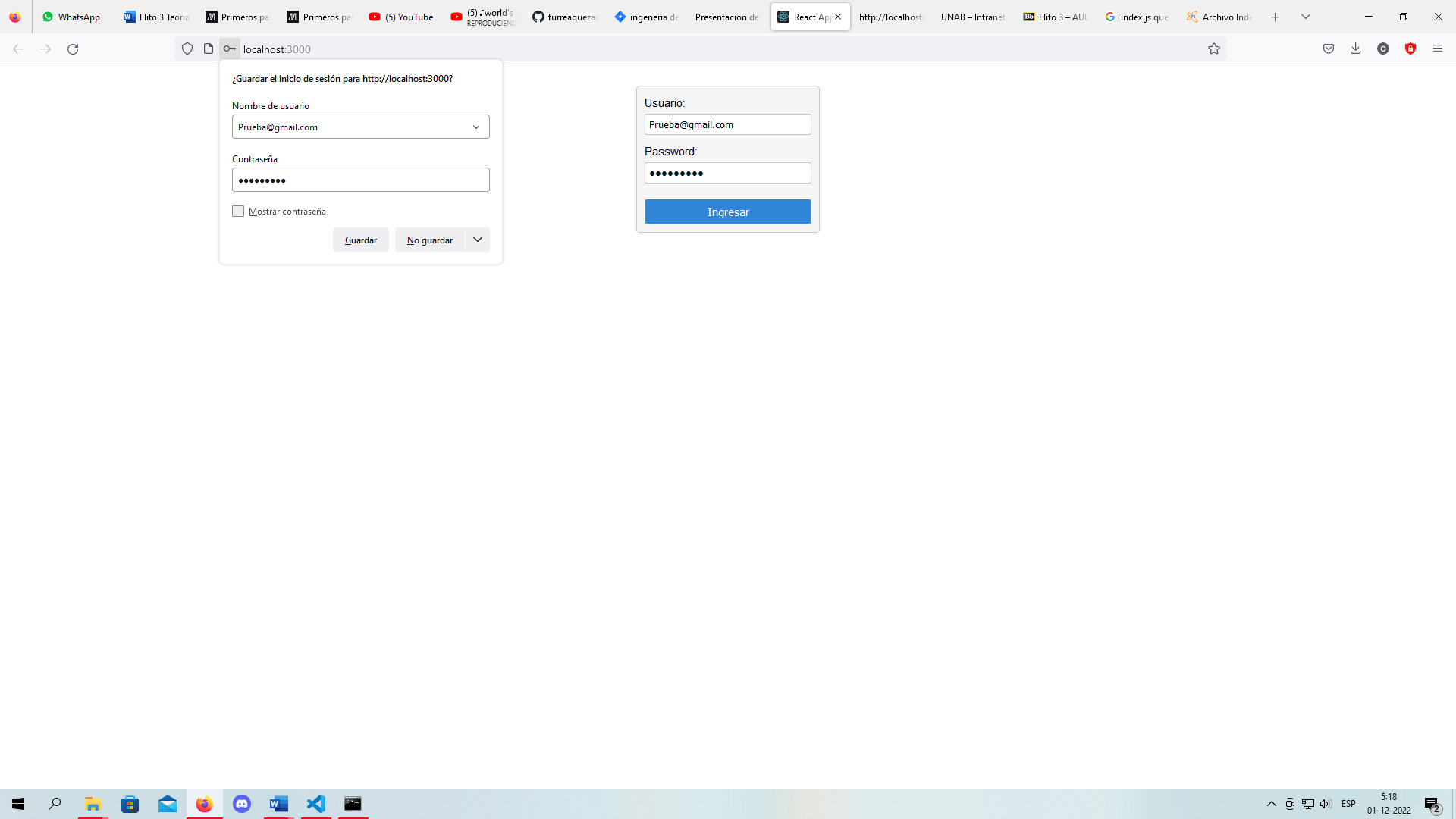




Login.css es el archivo que contiene los estilos del login.js solo se le pusieron códigos para centralizar la información.



Vista del login



# Conclusiones

# Referencias

GitHub Repositorio E-work

* <https://github.com/furreaquezada/ProyectoE-work>

Jira Proyect

* <https://searchwork.atlassian.net/jira/software/projects/ISW1/boards/1/backlog>

Pruebas de testing

* <https://programacionymas.com/blog/tipos-de-testing-en-desarrollo-de-software>

CSIRT

* <https://www.csirt.gob.cl/media/2022/01/INFORME-ANUAL-2021-CSIRT-de-Gobierno.pdf>

Lean Inception

* <https://martinfowler.com/articles/lean-inception/>

## Docker

-Requisitos Docker

* <https://babel.es/es/Media/Blog/Abril-2017/Docker-en-Windows-10>

SonarQube

-Métricas SonarQube

* <https://oscarmoreno.com/metricas-de-sonarqube-i/>
* <https://enmilocalfunciona.io/las-tripas-de-sonarqube-metricas-y-como-calcula-el-rating-parte-2/>

-Instalación SonarQube

* <https://testeandosoftware.com/sonarqube-instalacion-basica/>
* <https://community.sonarsource.com/t/installing-sonar-scanner-in-alpine-linux-docker/7010/5>
* <https://www.youtube.com/watch?v=vyZYMD8naB8>

-Uso SonarQube y Sonar Scanner

* <https://www.youtube.com/watch?v=cOQHc_-CjA4>

-Funcionamiento de Sonar Scanner

* <https://docs.sonarqube.org/latest/analysis/scan/sonarscanner/>