Texto

Descripción generada automáticamente

**AUTOMATIZACION DE PRUEBAS Y CONTROL DE CALIDAD BASADO EN PLAYWRIGHT Y BDD**

**UNIVERSIDAD NEBRIJA GRADO EN**

**INGENIERÍA INFORMÁTICA**

**MEMORIA TRABAJO EVALUACIÓN DE CAPACIDADES EN LA EMPRESA**

**José María Fernández Gómez**

**19/09/2024**

Texto

Descripción generada automáticamente

**AUTOMATIZACION DE PRUEBAS Y CONTROL DE CALIDAD BASADO EN PLAYWRIGHT Y BDD**

**UNIVERSIDAD NEBRIJA GRADO EN**

**INGENIERÍA INFORMÁTICA**

**MEMORIA TRABAJO EVALUACIÓN DE CAPACIDADES EN LA EMPRESA**

**José María Fernández Gómez**

**19/09/2024**

Adolfo Abalo Cascallar

D. /Dña. José María Fernández Gómez autoriza a que el presente trabajo se  
guarde y custodie en los repositorios de la Universidad Nebrija y además SI autoriza a su  
disposición en abierto

[1. RESUMEN 6](#_Toc184972395)

[2. PRÁCTICAS. 6](#_Toc184972396)

[2.1.1. ¿Qué entrevistas he hecho? 6](#_Toc184972397)

[2.1.2. Datos de la empresa. 7](#_Toc184972398)

[2.1.3. Información Previa (asignaturas, cursos previos, conocimiento previo, etc.). 7](#_Toc184972399)

[2.1.4. Motivos para elegir estas prácticas 7](#_Toc184972400)

[2.2. Objetivos de las prácticas 7](#_Toc184972401)

[2.3. Onboarding 7](#_Toc184972402)

[2.3.1 Cursos y Formaciones 8](#_Toc184972403)

[CURSO 1: Gherkin Language 8](#_Toc184972404)

[CURSO 2: Playwright: 9](#_Toc184972405)

[CURSO 3: Anti-Corrupción 9](#_Toc184972406)

[CURSO 4: prevención de riesgos laborales 10](#_Toc184972407)

[CURSO 5: Integridad 10](#_Toc184972408)

[CURSO 6(dos): World climate change e Igualdad 11](#_Toc184972409)

[CURSOS internos de Deloitte: 11](#_Toc184972410)

[2.4. Funciones y tareas en las prácticas. (evidencias) 11](#_Toc184972411)

[2.4.1 Test de rendimiento 12](#_Toc184972412)

[2.4.2 Test Funcional 12](#_Toc184972413)

[2.4.2.1 Bug 12](#_Toc184972414)

[2.4.2.2 Defecto 13](#_Toc184972415)

[2.4.2.3 Error 13](#_Toc184972416)

[2.4.3 Test Automatizado 13](#_Toc184972417)

[2.4.4 Desarrollo y creación del test plan 13](#_Toc184972418)

[2.5. Relaciones de problemas planteados y procedimientos para su resolución. 13](#_Toc184972419)

[2.6. Aprendizajes y desarrollo profesional (habilidades adquiridas) 14](#_Toc184972420)

[2.8. Herramientas utilizadas en las prácticas (evidencias). 14](#_Toc184972421)

[2.9. Logros, resultados y discusión 17](#_Toc184972422)

[1. Entendimiento del ciclo de vida de proyectos tecnológicos 17](#_Toc184972423)

[2. Agilidad y metodologías de trabajo 17](#_Toc184972424)

[3. Herramientas de gestión y colaboración 17](#_Toc184972425)

[4. Análisis y documentación de requisitos 17](#_Toc184972426)

[5. Desarrollo de habilidades interpersonales y comunicación profesional 18](#_Toc184972427)

[6. Organización personal y priorización de tareas 18](#_Toc184972428)

[7. Cumplimiento de plazos de entrega 18](#_Toc184972429)

[8. Integración de herramientas de automatización 18](#_Toc184972430)

[Playwright 18](#_Toc184972431)

[Puppeteer 18](#_Toc184972432)

[Selenium 19](#_Toc184972433)

[9. Comparativa entre frameworks 19](#_Toc184972434)

[3.1.1. Motivación 20](#_Toc184972435)

[3.1.2. Análisis de mercado y necesidades 20](#_Toc184972436)

[3.1.2.1 Aumento en la Complejidad y Escalabilidad del Software 20](#_Toc184972437)

[3.1.2.2 Exigencias de Calidad y Satisfacción del Cliente 20](#_Toc184972438)

[3.1.2.3 Reducción de Costes a Largo Plazo 20](#_Toc184972439)

[3.1.2.4. Competencia y Adopción por Parte de los Líderes del Mercado 21](#_Toc184972440)

[3.1.3. Objetivos 21](#_Toc184972441)

[3.1.4. Requisitos técnicos 21](#_Toc184972442)

[3.1.5. Análisis de mercado 27](#_Toc184972443)

[3.2. Marco técnico 27](#_Toc184972444)

[3.3. Equipo de trabajo y metodología 27](#_Toc184972445)

[3.4. Proyecto 27](#_Toc184972446)

[3.4.1. Resumen de contribuciones y productos 27](#_Toc184972447)

[Aplicativo de Pruebas en Entorno de Desarrollo: 27](#_Toc184972448)

[Scripts de Pruebas Automatizadas para el Entorno de Producción: 27](#_Toc184972449)

[Scripts de Ejecución para Diferentes Casos de Prueba: 27](#_Toc184972450)

[3.4.2. Planificación temporal 28](#_Toc184972451)

[3.4.3. Recursos empleados 30](#_Toc184972452)

[3.4.4. Trabajo desarrollado 30](#_Toc184972453)

[Referencias 30](#_Toc184972454)

## 1. RESUMEN

Alumno José María Fernández Gómez del grado Ingeniería Informática realizando las prácticas curriculares en la empresa Deloitte Technology & Transformation en el departamento de Quality Engineering para el proyecto tecnológico del

IOC(International Olympic Commitee). Realizando tareas desde lacreación del test plan para aplicativos individuales, ejecución de dicho test plan, hasta las pruebas end to end de todo el ecosistema de aplicativos que rodea al evento de los Juegos Olímpicos.

## 2. PRÁCTICAS.

### 2.1.1. ¿Qué entrevistas he hecho?

1 AÑO

* Proceso de selección en Deloitte: No seleccionado en última fase
* Proceso de selección en MET Life: No seleccionado por incompatibilidad de horarios
* Proceso de selección en BBVA: No seleccionado por incompatibilidad de horarios

2 AÑO

* Proceso de selección Abaqo: Rechazado por ser oferta de media jornada
* Proceso de selección BBVA: No seleccionado por motivos desconocidos
* Proceso de selección NTT Data: Seleccionado
* Proceso de selección Deloitte: Seleccionado

Se hicieron procesos en dos años ya que el alumno decidió decidió no cursar las prácticas curriculares debido a la incompatibilidad de horarios con su rehabilitación neurológica .

Con respecto a los dos procesos seleccionados en el segundo año, Deloitte

y NTT Data, se tuvo que tomar una decisión tomando en cuenta diversos aspectos que posteriormente mencionaré, pero se acabó optando por la consultora Deloitte.

Se eligió Deloitte frente a NTT Data principalmente por el alcance del proyecto al que el alumno iba a ser asignado, frente a la consultora japonesa, NTT data, que no proporcionó información sobre el proyecto al que en principio el alumno fuera a ser asignado.

### 2.1.2. Datos de la empresa.

Deloitte, en particular Deloitte Technology & Transformation es una empresa dedicada a la consultoría tecnológica, como su propio nombre indica, pende de la consutora Deloitte Touche Tohmatsu Limited, la cual forma parte del grupo conocido como “Big Four”, dentro de este grupo en particular, se caracteriza por ser una de las más punteras y enfocadas en el área de tecnología.

Posee distintas líneas de negocio entre las que se encuentran:

* **Estrategia y Arquitectura Tecnológica:**
* **Arquitectura e Innovación:**
* **Valor del Dato**
* **Fusiones y Adquisiciones (M&A)**

(Deloitte, s.f.)

### 2.1.3. Información Previa (asignaturas, cursos previos, conocimiento previo, etc.).

Sobre la empresa en cuestión simplemente se conocía la existencia de esta y el peso a nivel internacional que, como gran consultora tenía.

### 2.1.4. Motivos para elegir estas prácticas

1. Reconocimiento Internacional: Deloitte es una empresa reconocida y reputada a nivel internacional, no únicamente en el ámbito de la tecnología.
2. Proyecto con perspectiva: El proyecto, al estar planificado para desplegarse en el año 2026 y 2028 tiene mucha perspectiva de futuro.
3. Oportunidades: Se ofreció la posibilidad de desarrollo dentro de la empresa y de cambio de departamento/línea de negocio si así se deseaba.
4. Condiciones de Presencialidad: Se ofreció una modalidad híbrida, con algunos días de obligatoria presencialidad
5. Proximidad de Oficinas: Las oficinas se encuentran cerca de mi lugar de residencia, por lo que se valoraba el tiempo y costo de desplazamiento como factor secundario
6. Condiciones laborales: Se ofrecieron unas condiciones horarias y salariales atractivas

## 2.2. Objetivos de las prácticas

* Desarrollo de las tareas bajo un marco de trabajo agile
* Aprendizaje de herramientas de gestión del proyecto como Jira, Confluence...
* Entendimiento de requisitos de negocio
* Comprobación del cumplimiento de requisitos de negocio
* Entendimiento de historias de usuario
* Automatización de tests para comprobar historias de usuario

## 2.3. Onboarding

El Proceso de onboarding duró un par de semanas, donde se me plantearon varios cursos y formaciones de carácter obligatorio. Todo el proceso comenzó a fecha de 16 de septiembre de 2024, día en el que se me indicó que debería ir a las oficinas y que un miembro del equipo me recibiría.

El miembro en cuestión fue Jorge Añez, el cual se introdujo, me dio una vuelta por las distintas plantas en las que estaría trabajando y posteriormente me indicó que debería realizar unas primeras tareas de onboarding donde se me entregaría el material de trabajo y algunos presentes de bienvenida.

Durante este periodo, también tuve que realizar unos primeros cursos, algunos para conocer a la firma y otros por temas legales y administrativos (firma de un NDA, cursos de “non-compliance...”).

Por otro lado, se me indicó que debería realizar de manera paralela un onboarding orientado al cliente, el Comité Olímpico Internacional. En este onboarding se me formó en torno a la historia, cultura y estructura jerárquica de dicha institución, aparte de indicar la estructura a nivel internacional y nacional de las distintas entidades que lo conforman.

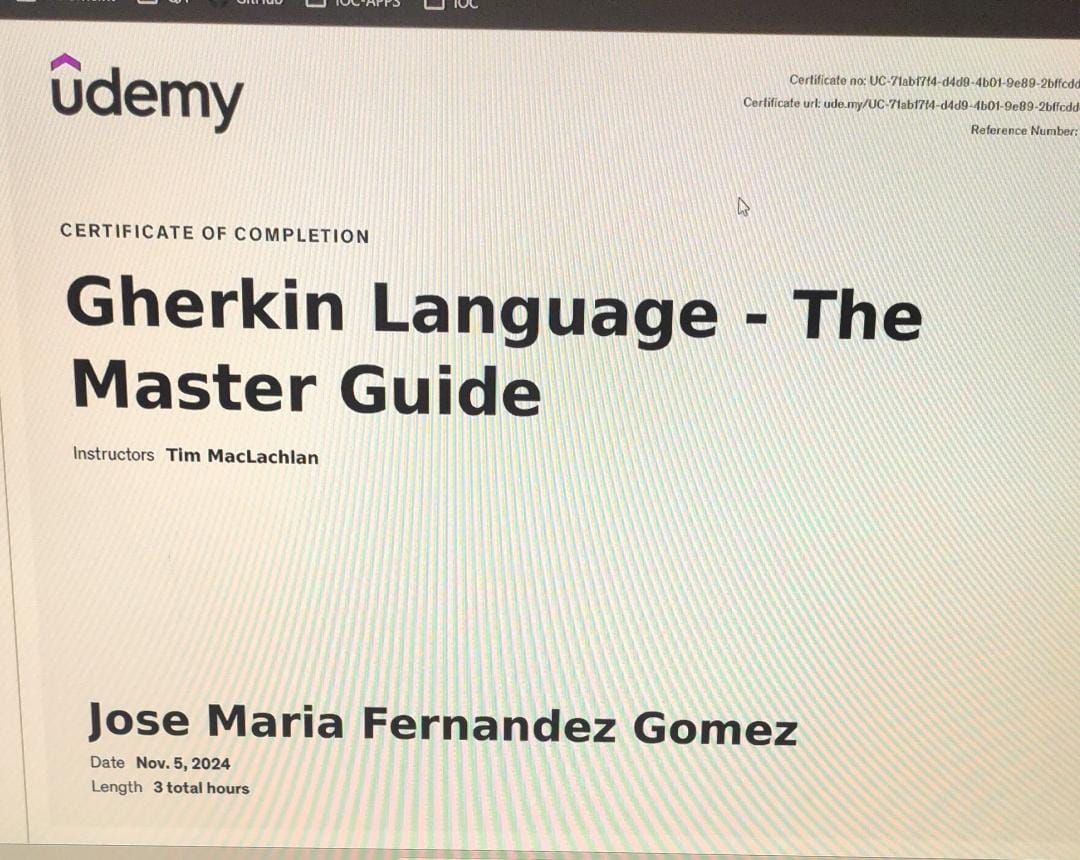
Por último, en el marco más técnico, se me dio una formación sobre el framework que utilizaríamos en las labores de QA y automatización de tests, esto se hizo a través de diversas formaciones online por el portal corporativo de Udemy. En estos cursos se mostraba cómo desplegar tests basados en historias de usuario, siempre llevando un formato determinado. Estos tests, podrían ser lanzados sobre aplicativos web, ya que trabajan con los componentes renderizados en una máquina cliente, aunque también se pueden comprobar las llamadas de red que el dispositivo cliente puede realizar, pero esto se explicará mejor en apartados futuros.

### 2.3.1 Cursos y Formaciones

Como parte del proceso de onboarding, se me asignaron distintos cursos y formaciones que debería tomar, muchos de ellos a nivel corporativo, gestión de riesgos y temas similares referidos a políticas internas de la empresa, y por otro lado, se me asignaron distintos cursos a nivel más técnico, para familiarizarme con las tecnologías con las que se iba a trabajar. A continuación, se muestran los cursos tomados y los certificados de consolidación de estos:

#### CURSO 1: Gherkin Language

* + Tecnología orientada al desarrollo guiado por comportamiento (BDD), posteriormente se hablará más sobre esta tecnología



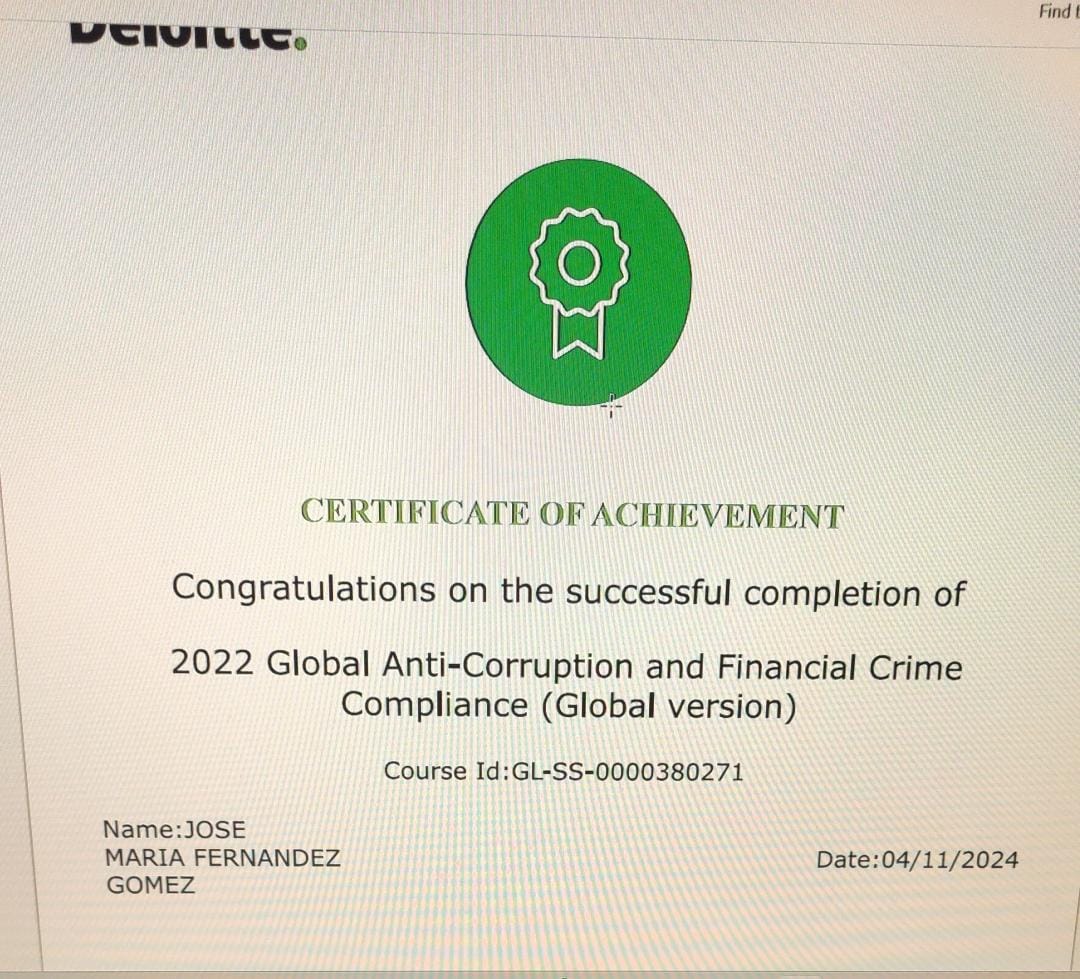
#### CURSO 2: Playwright:

* Tecnología empleada para la automatización de tests sobre aplicativos web basada en typescript y NodeJs



#### CURSO 3: Anti-Corrupción

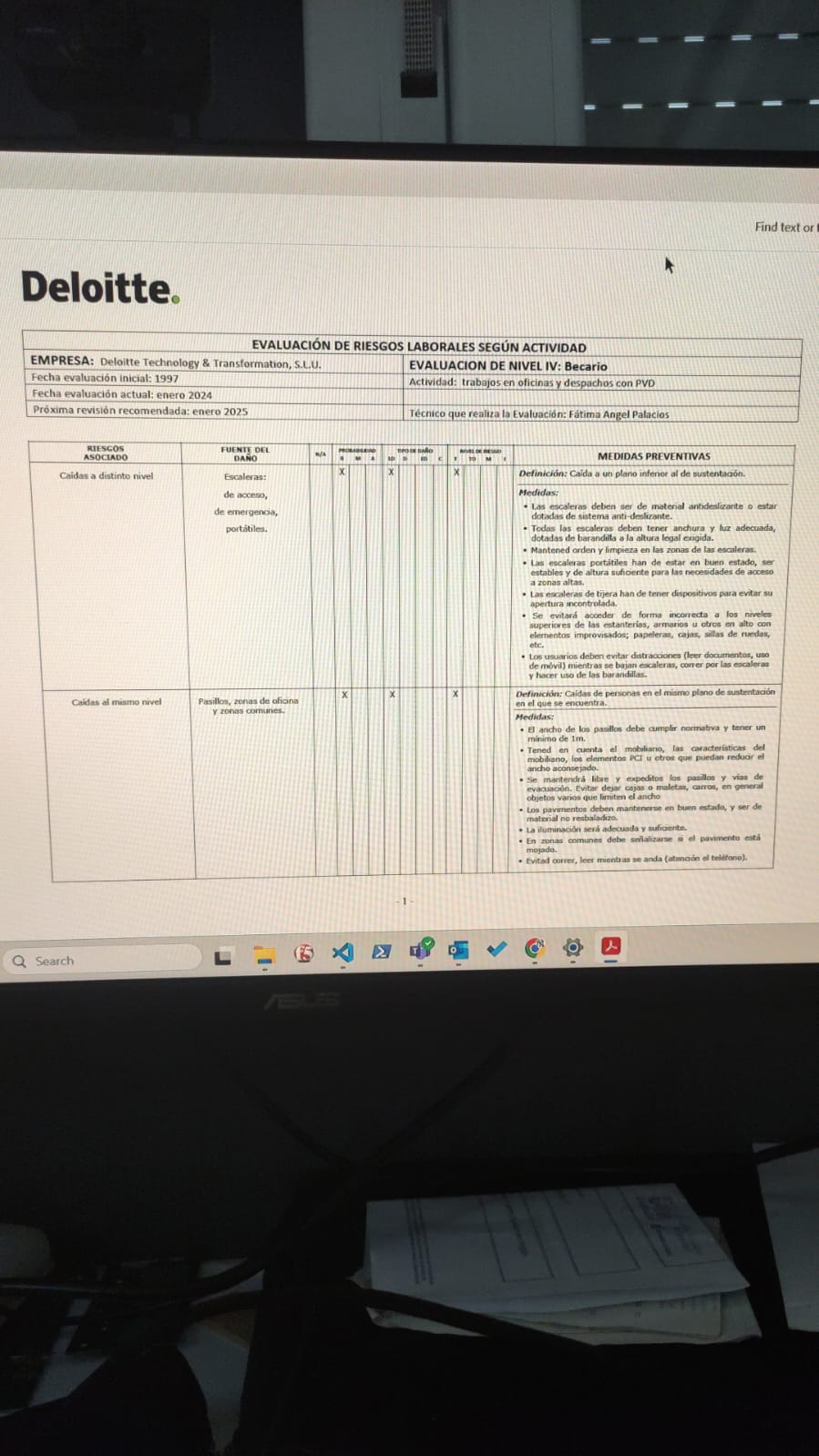
Curso interno de Deloitte para garantizar el cumplimiento de la ley independientemente del proyecto o producto asignado.



TEST 456 2

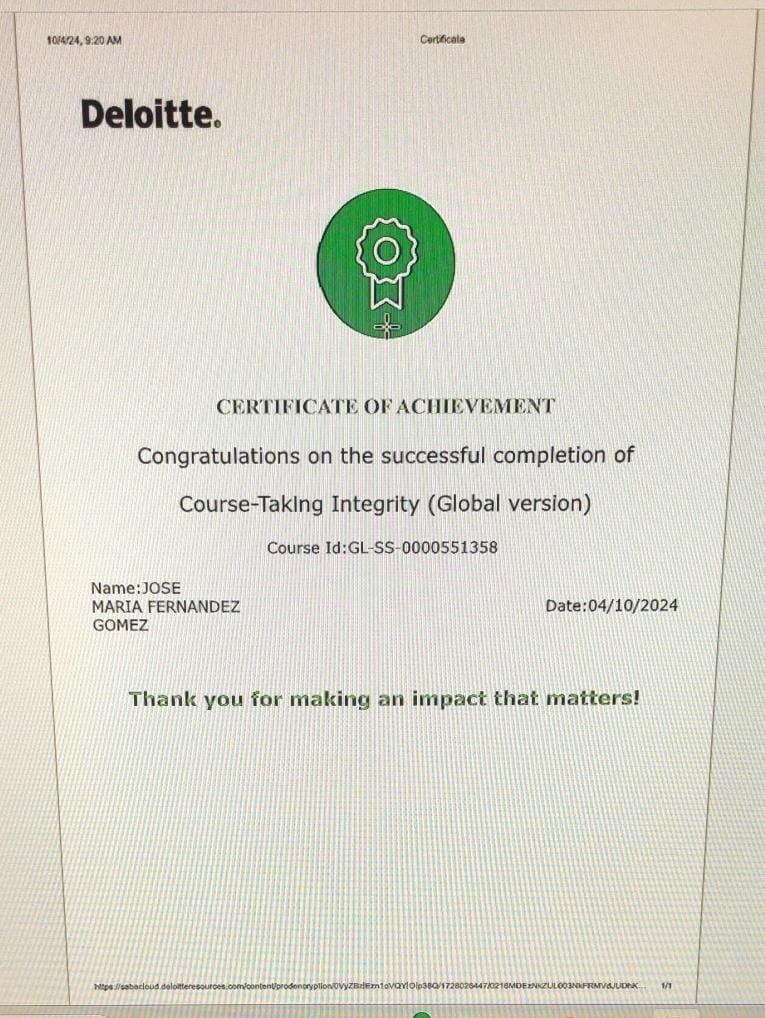
#### CURSO 4: prevención de riesgos laborales

Curso interno de Deloitte para garantizar el cumplimiento de la ley independientemente del proyecto o producto asignado.



#### CURSO 5: Integridad

Curso interno de Deloitte para garantizar la independencia del trabajo y servicio asociado independientemente de las relaciones personales/profesionales/privadas



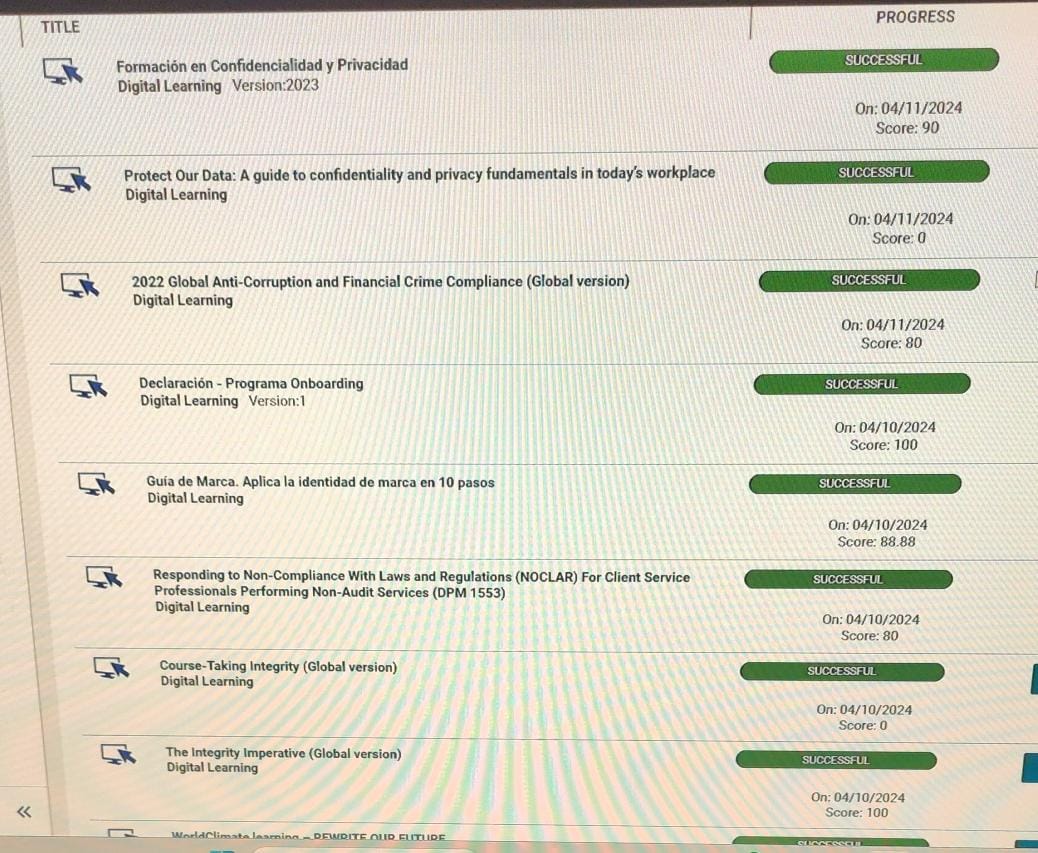
#### CURSO 6(dos): World climate change e Igualdad

Curso interno de Deloitte para concienciar sobre el tema del cambio climático y la igualdad en el ámbito laboral.

No se posee foto de este ya que la certificación caducó al poco de hacerlo

#### CURSOS internos de Deloitte:

Anteriormente se han puesto como ejemplo dos formaciones internas obligatorias de la empresa, pero se tuvieron que realizar bastantes más, abajo quedan marcadas los certificados de aprobación de todos



## 2.4. Funciones y tareas en las prácticas. (evidencias)

Por motivos de seguridad de la empresa Deloitte, no se me permite sacar código ni capturas de pantalla sobre los aplicativos que se está trabajando. Lo que sí puedo afirmar es que se están realizando tareas muy similares a las propuestas en la memoria de desarrollo, sobre aplicativos propietarios del IOC, los cuales algunos pertenecen a empresas externas a Deloitte.

A nivel informativo indicaré los tres tipos de tareas que me han encargado hacer.

### 2.4.1 Test de rendimiento

Las pruebas de rendimiento se desarrollan en un entorno de integración, donde el aplicativo a testear ya han superado pruebas previas como la de seguridad o las pruebas funcionales. Suelen consistir en la simulación de usuarios accediendo al sistema de manera concurrente, realizando tareas esperadas para el sistema como podrían ser la autenticación, navegación por las páginas y en función de cada aplicativo, desempeñar las tareas para que fueran diseñados.

### 2.4.2 Test Funcional

“El testing funcional de un aplicativo en el ámbito de Quality Assurance (QA) se refiere al proceso de verificar que las funcionalidades del software funcionan según lo especificado en los requisitos. En otras palabras, este tipo de pruebas evalúa si el sistema realiza correctamente las tareas y funciones para las que fue diseñado, centrándose en el "qué" hace el software, en lugar de "cómo" lo hace.”

ChatGPT extracted from: Foundations of Software Testing" de Dorothy Graham y "Software Testing: Principles and Practices" de Srinivasan Desikan y Gopalaswamy Ramesh.

En otras palabras, el test funcional se refiere a la comprobación a alto nivel de las funcionalidades que un usuario objetivo del aplicativo debería ser capaz de ejecutar.

Algunos ejemplos de un test funcional podrían ser:

a. Comprobación de la funcionalidad de reservas en el aplicativo de un restaurante:

1. Acceder al sitio web desde un ordenador común

2. Hacer click en la pestaña de reservas

3. Intentar reservar un día que está completo

4. Esperar un mensaje de error indicando que no está disponible

5. Intentar reservar un día con disponibilidad

6. Esperar una respuesta satisfactoria

7. Esperar un email de confirmación

b. Comprobación de la funcionalidad de subida de documentos en un back office:

1. Acceder al aplicativo de back office

2. Hacer click en la pestaña de uploads

3. Hacer click en new

4. Añadir un archivo PDF

5. Hacer click en subir

6. Acceder al aplicativo de front office

7. Hacer click en la pestaña de files

8. Comprobar que el archivo PDF se ha subido correctamente

Esta es la primera fase dentro del cumplimiento de un test plan, ya que se puede hacer de manera inmediata tras el despliegue de un aplicativo, y siempre se puede volver a realizar dichos tests.

Este tipo de tests, pueden ser realizados por personas con perfil no técnico, ya que se trata d, a alto nivel, tratar de ejecutar funcionalidades que el aplicativo debería poseer, como si de un usuario final se tratara.

En esta fase del proceso de test, es donde más bugs, defectos o errores de integración se suelen encontrar.

#### 2.4.2.1 Bug

Es una falla o anomalía en el comportamiento del programa debido a un error en el código o en el diseño. En general, se refiere al problema detectado en el software durante su ejecución o pruebas.

#### 2.4.2.2 Defecto

Es una desviación entre el comportamiento esperado del software y el comportamiento real que se

observa. Un defecto suele ser el resultado de un error en el código o en la especificación.

#### 2.4.2.3 Error

: Es una equivocación o fallo cometido por un desarrollador al escribir el código o diseñar el sistema. Surge de un malentendido, un descuido o una falta de conocimiento.

### 2.4.3 Test Automatizado

Los tests automatizados, se realizan también en entornos de integración, muy cercanos a lo que se supone que ha de ser el despliegue en producción del aplicativo en cuestión. En esta parte del proceso de validación de un aplicativo, es indispensable que haya ciertos valores, aspectos o funcionalidades del aplicativo que no vayan a ser modificadas en el paso de integración a producción, ya que la principal meta de estos tests es su replicabilidad, el poder ser lanzados y los resultados sean consistentes cada vez que se ejecutan. En este apartado es donde se centrará el desarrollo principal del proyecto que acompaña esta memoria. En concreto, se planifica el desarrollo y despliegue de test sobre aplicativos web, donde se pretenderá simular el comportamiento de un usuario final de distintos aplicativos por medio del uso del lenguaje Gherkin, lenguaje conocido por ser una metodología de testing que une la parte de Ingeniería del software y testing técnico. El lenguaje Gherkin es un nexo perfecto entre los dos ámbitos de la gestión de un proyecto tecnológico ya que permite la aplicación directa de las historias de usuario, previamente extraídas de una toma de requisitos, en nuestro testing automatizado.

### Desarrollo y creación del test plan

El test plan es una lista interna dentro del departamento de Quality Assurance en el cual se detalla qué está y qué no está dentro del scope de la fase de pruebas, ya sea para una determinada release de código, o a la hora de poder probar un aplicativo completo, detallando todos los flujos posibles, ya sean intencionados o no, de dicho aplicativo.

En esta etapa de la fase de pruebas se trata de organizar los requisitos de negocio tomados directamente al cliente, producto owner o persona encargada/interesada, para poder ser capaces de agilizar, reportar y dejar documentadas cada una de las pruebas que se han realizado.

Este test plan, como se ha mencionado previamente, ha de ir estrechamente ligado a la toma de requisitos, paso indispensable en el proceso de garantizar la calidad de un producto software.

## 2.5. Relaciones de problemas planteados y procedimientos para su resolución.

1. Problema con la sim proporcionada:

* Solución: La tarjeta sim proporcionada en el onboarding no funcionaba correctamente, y se me instruyó en cómo abrir un ticket en serviceNow para labores de este estilo

1. Falta de permisos para acceder a productos de Atlassian:

* Solución: Se tuvo que abrir una incidencia en serviceNow y solicitar la aprobación del socio responsable

1. TO BE IMPLEMENTED

2.6. Aprendizajes y desarrollo profesional (habilidades adquiridas)  
2.7. Metodologías utilizadas.

Se trabaja bajo un marco de trabajo Agile, en concreto con metodología Scrum, ya que todas las semanas se tiene una reunión "weekly” para analizar el progreso hacia el objetivo del sprint actual, y cada equipo en su conjunto tiene reuniones “daily”, en concreto, en el departamento de QA, al depender directamente del producto entregado por los equipos de desarrollo, ya sean propios internos de Deloitte o proveedores externos.

Por otro lado, para la automatización de tests se trabaja bajo un marco “Behavior Driven Developement” donde tanto el equipo de producto, como el equipo de desarrollo, como el equipo de testing, se han de poner de acuerdo a la hora de definir unos requerimientos sobre los que cada uno realizar su labor. La labor del equipo de producto sería realizar esos requisitos y dejarlos plasmados, para que posteriormente el equipo de desarrollo fuera capaz de crear un aplicativo, producto, servicio... y por último lugar, el equipo de Quality fuera capaz de comprobar esos requerimientos. Como se ha descrito previamente, la metodología BDD, es capaz de unificar esas tres áreas del producto, facilitando así la comunicación entre equipos y reduciendo los fallos de comunicación entre estos.

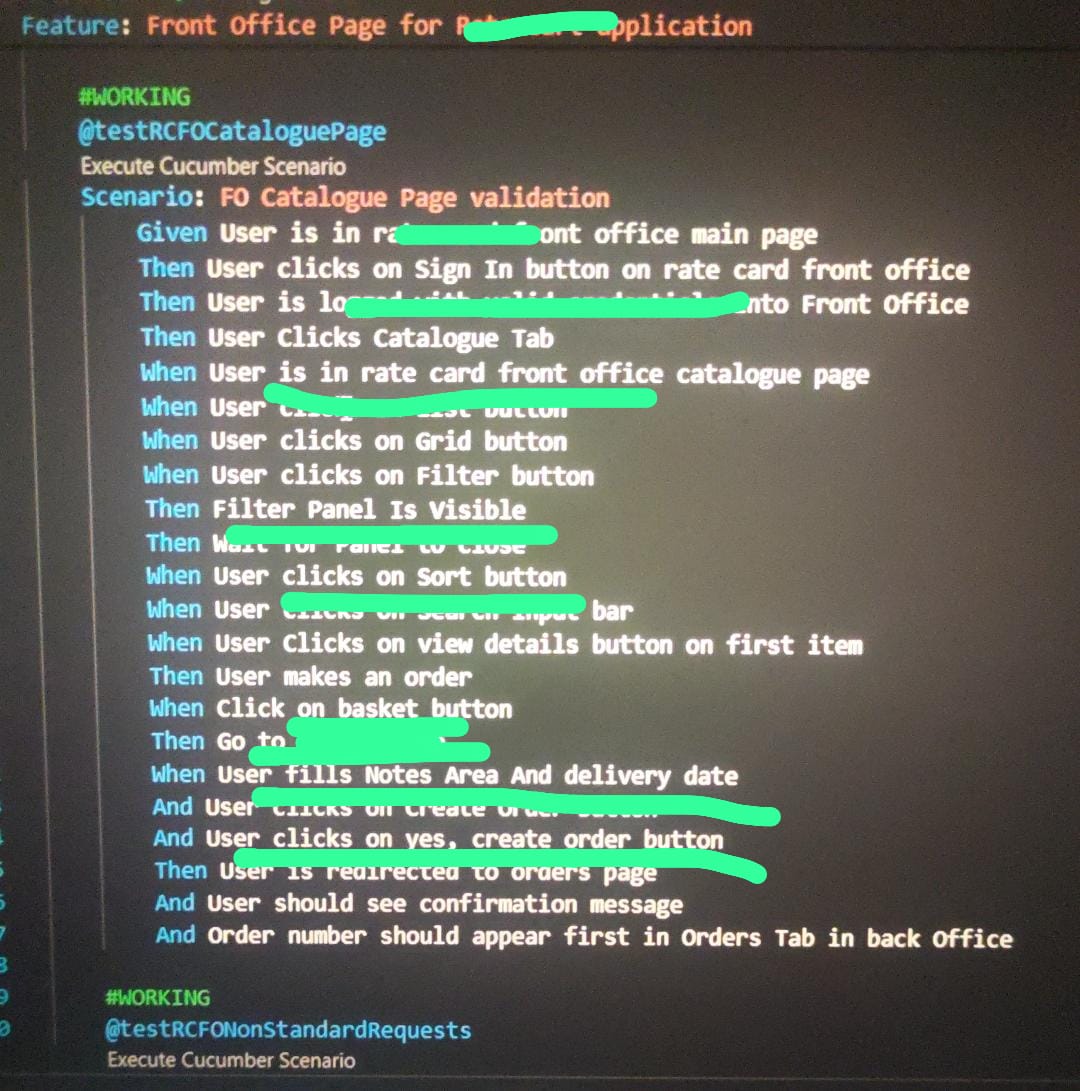
Por otra parte, tenemos que asistir a muchas ceremonias relacionados con otras empresas externas que también trabajan bajo un marco Agile, esto principalmente debido a que cuando estas hacen una release de un aplicativo que nuestro equipo está asignado para testear, se tiene que asistir a dicha ceremonia para poder entender y sintetizar bien todas las nuevas funcionalidades o “Features” que se nos entrega, funcionalidades que posteriormente deberemos abstraer para poder incorporarlas a nuestro banco de pruebas o “test plan”.

Las metodologías serán actualizadas más adelante, ya que aún hay varias fases del proceso de QA que no se han experimentado

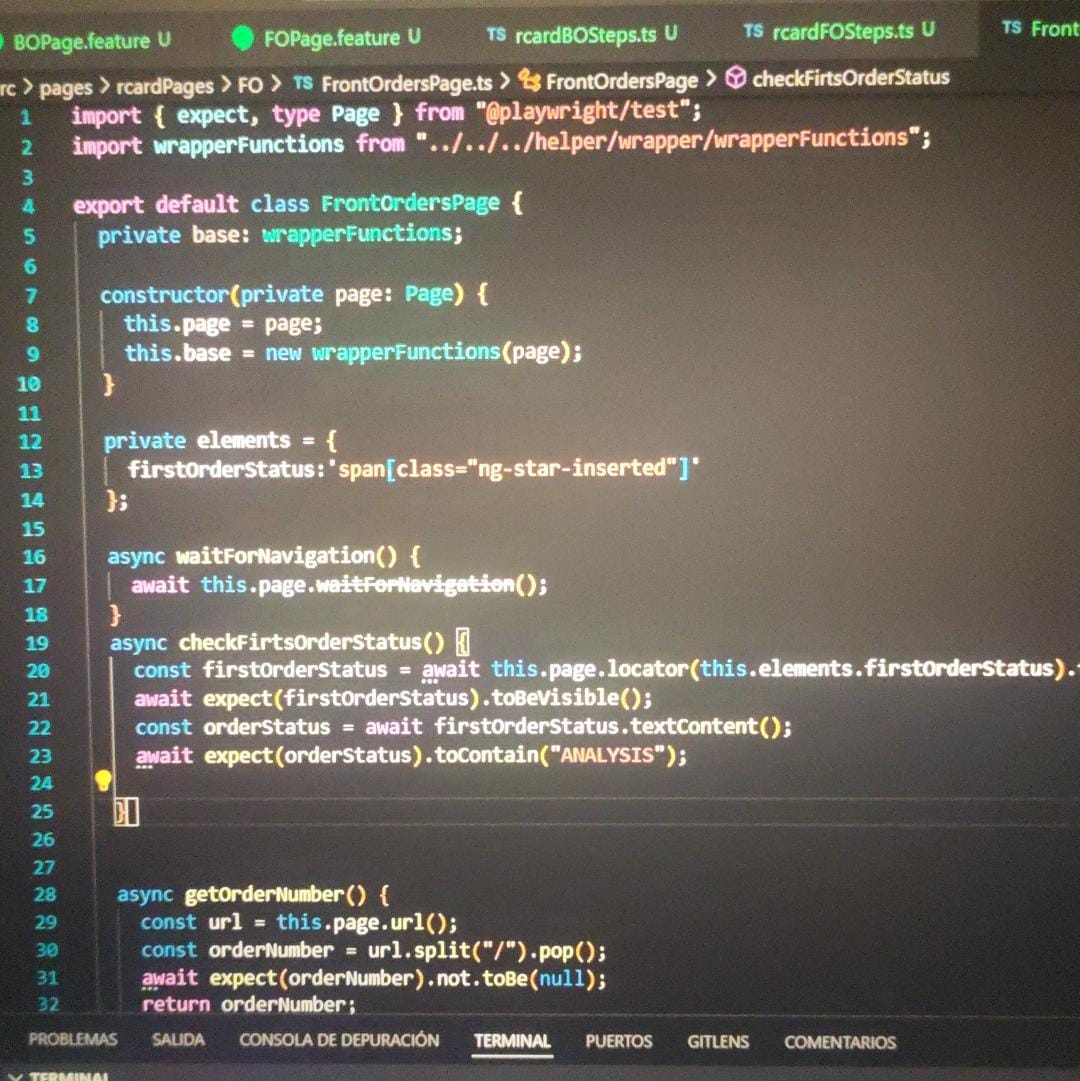
2.8. Herramientas utilizadas en las prácticas (evidencias).

* Cucumber for BDD

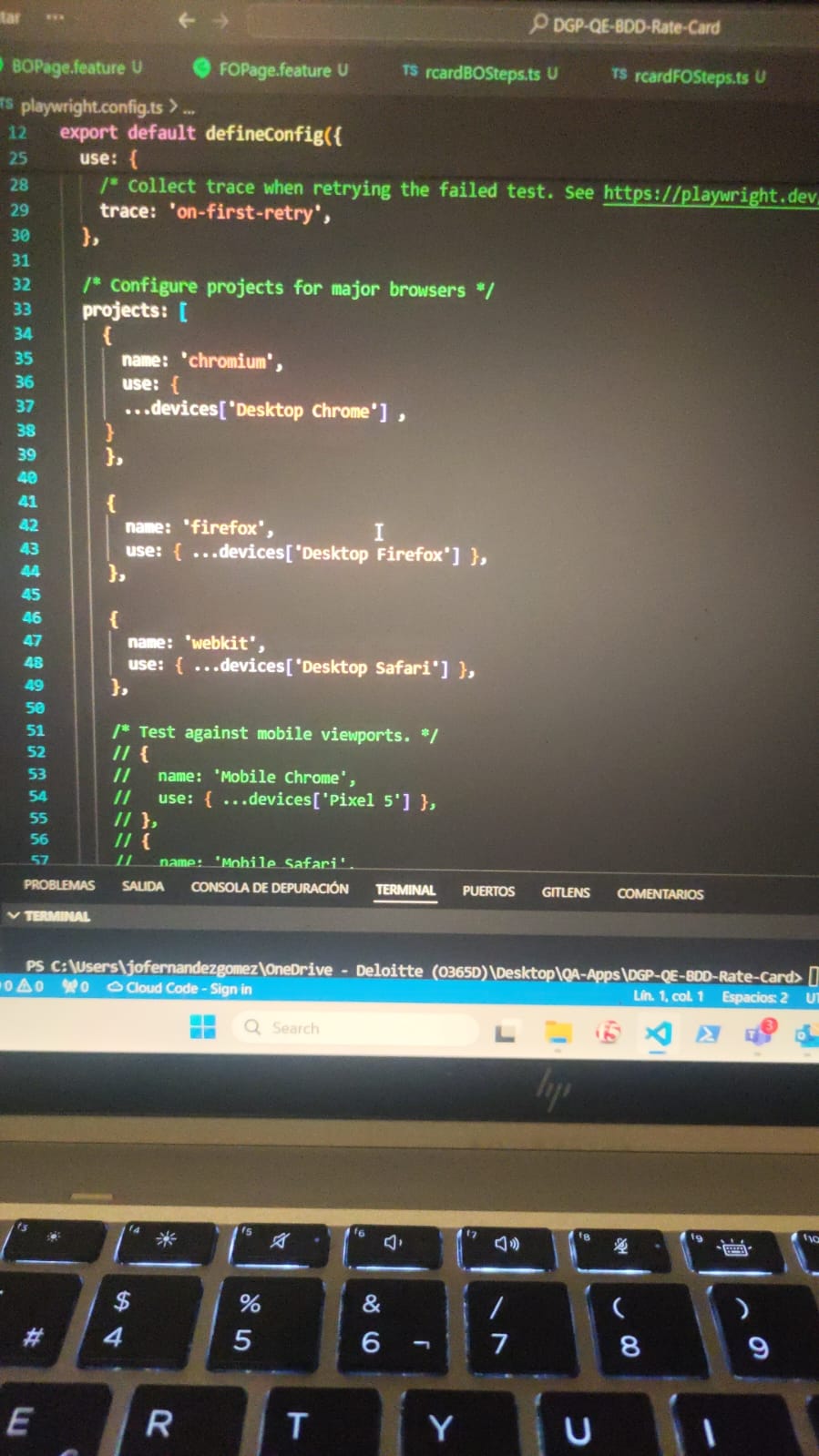
Cucumber es una herramienta que soporta el desarrollo guiado por comportamiento (Behavior Driven Development o BDD), un enfoque de desarrollo de software que promueve la colaboración entre desarrolladores, testers y el negocio. Cucumber permite escribir especificaciones funcionales de una aplicación en un lenguaje natural entendible tanto por personas técnicas como no técnicas, como **Gherkin**.



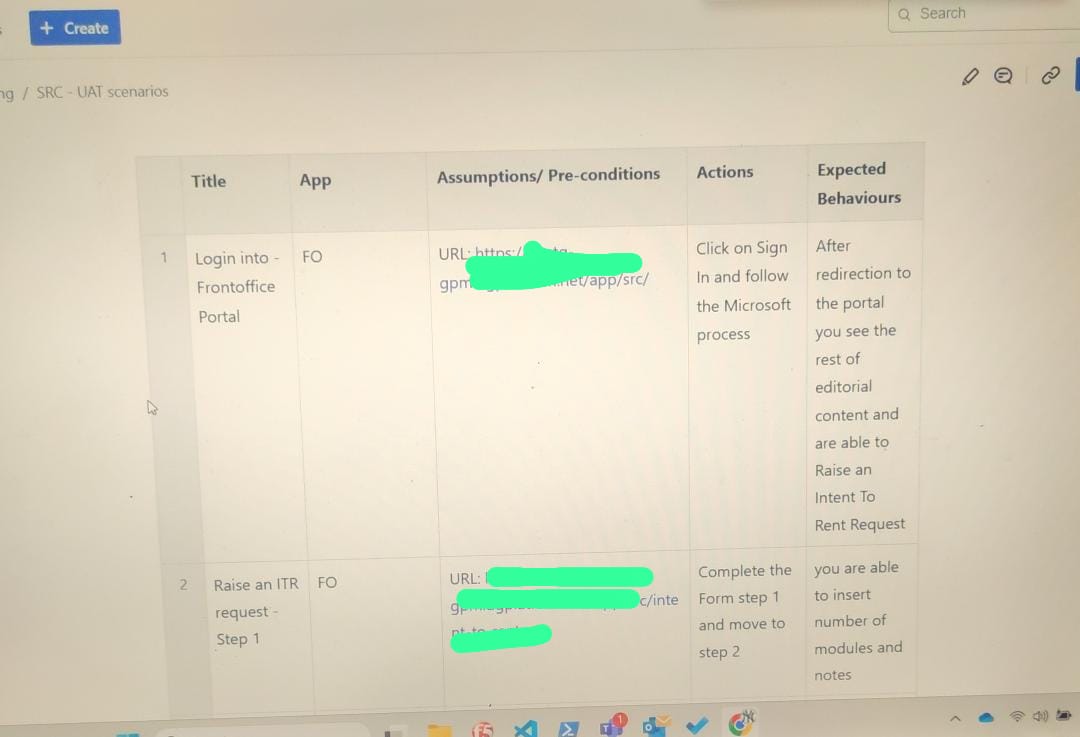
* NodeJs: NodeJs es el lenguaje de programación base donde corre todo el código a ejecutarse por los tests, esto es idóneo ya que estamos tratando con aplicativos webs, lo que facilita la integración de este framework con la interacción en los programas clientes. Se usa NodeJs por su velocidad trabajando con aplicativos web y por su facilidad a la hora de construir objectos, que es muy útil para poder tener estructurado todas las páginas del aplicativo encapsuladas en objetos.



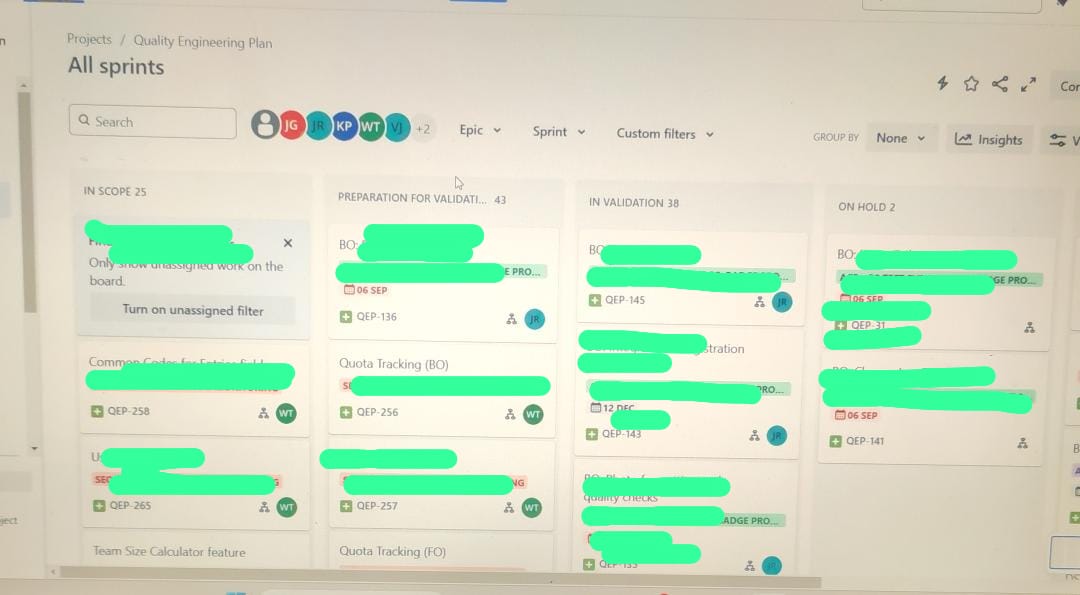
* Playwright



* Python: Usado para crear scripts rápidos y en especial para el tratamiento, formateo y presentación de datos
* Aplicativos web desarrollados por proveedores externos
  + Por motivos de confidencialidad, no se me permite adjuntar fotos de aplicativos externos.
* Microsoft teams para la comunicación
  + Con diversos grupos, tareas automatizadas y gestión de eventos a nivel corporativo
* VPNs para asegurar el acceso seguro a aplicativos en entorno de integración y desarrollo
* Suite de Atlassian
  + Confluence: En Confluence, se separa cada aplicativo a testear en espacios de trabajo distintos, donde se suben los requerimientos de las aplicaciones a testear:



* + Jira: En Jira, se tienen separadas por tiempos y por estados, todas las tareas que están dentro del “scope” del proyecto(o aplicativo)



## 2.9. Logros, resultados y discusión

#### 1. Entendimiento del ciclo de vida de proyectos tecnológicos

En una primera instancia, y a un nivel aún no extremadamente elevado, se ha comprendido como los productos tecnológicos tienen su ciclo de vida estándar, y como de manera iterativa pasan una y otra vez por las manos de los mismos equipos, generando así pequeñas modificaciones que ayuden a mejorar la calidad del producto.

Desde el punto de vista de un proyecto tan grande, el cual no solo involucra distintos productos de software, si no que también involucra la colaboración entre distintas empresas, poseyendo cada una distintos aplicativos software, que finalmente deberán ser testeados por el equipo de Quality de Deloitte. Esto en muchas ocasiones supone barreras, ya sean de lenguaje, horarias e incluso culturales, puesto que las distintas culturas, sociedades poseen distintos calendarios y franjas horarias de trabajo.

#### 2. Agilidad y metodologías de trabajo

Aplicación práctica de metodologías ágiles (como Scrum) para gestionar tareas diarias y participar en reuniones de planificación (sprints, dailies, retrospectivas) y como de vez en cuando pueden implicar un aumento en la calidad del producto y un entorpecimiento en el ciclo de vida del proyecto.

#### 3. Herramientas de gestión y colaboración

Uso de herramientas de gestión como Jira para la asignación y el seguimiento de tareas, y Confluence para documentar conocimientos y procesos.

Familiarización con plataformas de trabajo colaborativo y comunicación como Microsoft Teams o Slack, adaptándose a las dinámicas de trabajo en un entorno híbrido

#### 4. Análisis y documentación de requisitos

interpretación y comprensión de requisitos de negocio y su traducción en historias de usuario, bases para el desarrollo y pruebas de software. Todo esto facilitado por la suite de programas de Atlassian

#### 5. Desarrollo de habilidades interpersonales y comunicación profesional

En particular, a la hora de interactuar con el equipo más cercano, ya que para la interacción con personas de mayor rango (Asociados, socios...) se mantiene un tono extremadamente formal, frente a la comunicación con el entorno cercano, donde es más distendido, pero aun así guardando las formas y la profesionalidad.

#### 6. Organización personal y priorización de tareas

En una corporación como Deloitte, se aprende de manera clara y concisa la jerarquía dentro de esta empresa, y por ende, y en función de quién sea la persona que te pida una tarea, se aprende rápido a priorizar unas sobre otras. Como suelen ser varias las tareas que se están realizando de manera simultánea, se ha de tener una buena organización personal, y esto se hace más fácil gracias a herramientas como Microsoft ToDo-List.

#### 7. Cumplimiento de plazos de entrega

Se ha logrado entender cómo, y porqué, los productos de software han de tener fechas de entrega tan relativamente estrictas, ya que la cadena de producción o ciclo de vida del desarrollo software debe tener una fase final donde el producto es aceptado y validado por personas, consumidores o entidades terceras que puedan simular, o incluso que sean el destinatario final.

Enel ciclo de vida de este proyecto a esa fase se le ha llamado UAT o UVT (User Acceptance Test o User Validation Test)

#### 8. Integración de herramientas de automatización

Aprendizaje sobre el uso de herramientas de automatización de pruebas (por ejemplo, Playwright, Postman) y cómo estas contribuyen a mejorar la eficiencia en la validación de productos tecnológicos.

La herramientas de automatización empleadas son muy variadas, se ha tratado con distintos frameworks como playwright, pupeteer o selenium. Todos estos frameworks proporcionan

Reflexión sobre la importancia de equilibrar las pruebas manuales y automáticas para maximizar la cobertura de pruebas sin comprometer la calidad.

##### Playwright

* Es un framework moderno que permite la automatización de navegadores como Chrome, Firefox y Safari, con un enfoque en la consistencia y la ejecución paralela.
* **Características principales**:
  + Soporte para múltiples lenguajes de programación como JavaScript, Python y C#.
  + Capacidad para manejar situaciones complejas, como múltiples pestañas, ventanas emergentes y manejo avanzado de autenticación.
  + Excelente integración con pruebas en entornos de CI/CD.
  + Uso común en pruebas end-to-end (E2E) debido a su rapidez y confiabilidad.

##### Puppeteer

* Este es un framework desarrollado por Google, principalmente enfocado en la automatización del navegador Chrome.
* **Características principales**:
  + Ideal para generar pruebas de regresión visual y scraping de datos web.
  + Ofrece un control detallado del navegador, lo que lo hace excelente para pruebas personalizadas y generación de flujos específicos.
  + Aunque es más limitado que Playwright en términos de soporte para navegadores, su simplicidad lo hace adecuado para casos donde solo se requiere Chrome.

##### Selenium

* Uno de los frameworks más antiguos y populares en el ámbito de la automatización de pruebas. Su versatilidad y amplia comunidad lo convierten en una opción confiable para proyectos con múltiples requerimientos.
* **Características principales**:
  + Soporte para múltiples navegadores y lenguajes de programación, incluyendo Java, Python, y C#.
  + Capacidad para ejecutar pruebas en entornos distribuidos gracias a Selenium Grid.
  + Integración con herramientas de CI/CD y su utilidad en proyectos legacy o que requieren compatibilidad amplia.

Fuentes: MarketingScoop (Marketing Scoop, 2024)

#### 9. Comparativa entre frameworksCaptura de pantalla con la imagen de una pantalla Descripción generada automáticamente con confianza media

ChecklyHQ, Bright Data, & Marketing Scoop. (2024). Comparativa de características de herramientas de automatización: Playwright, Puppeteer y Selenium. Adaptado y sintetizado por ChatGPT, diciembre 2024. Recuperado de https://www.checklyhq.com, https://brightdata.es, https://www.marketingscoop.com.

3. DESARROLLO

3.1. Introducción

Para la parte de desarrollo dentro de esta tarea, se van a plantear dos escenarios, uno sobre un entorno de desarrollo/integración, representado por un aplicativo web propio que alumno desarrollará en Python con múltiples funcionalidad, que, aunque no sean del todo coherentes entre sí dentro de un entorno realista, servirá para demostrar todo el potencial que el framework y las tecnologías del desarrollo de tests puede realizar, por otro lado, se pretende hacer una batería de tests sobre aplicativos de la propia universidad, un entorno ya de producción, y, por lo que esto implica, un entrono robusto a ataques automatizados, entre los cuales podrían catalogarse los scripts que conforman este banco de pruebas.

3.1.1. Motivación

Con el ánimo de demostrar todo lo que este framework es capaz de hacer, se ha decidido hacerlo sobre dos entornos destinitos, como se ha explicado previamente, uno de producción y otro de integración/desarrollo. Esto con el ánimo de mostrar todo lo que una buena planificación de tests, unidos a una buena toma de requisitos y una implementación correcta de los tests es capaz de producir.

### 3.1.2. Análisis de mercado y necesidades

#### 3.1.2.1 Aumento en la Complejidad y Escalabilidad del Software

A medida que las aplicaciones se vuelven más complejas y distribuidas, con microservicios, APIs y múltiples plataformas, el número de casos de prueba y la complejidad de las pruebas aumenta. Las pruebas manuales no son sostenibles en este contexto.

La automatización de pruebas permite escalar las pruebas de manera eficiente y cubrir una mayor variedad de escenarios, plataformas y dispositivos, lo cual es crítico para aplicaciones que deben operar en entornos heterogéneos.

#### 3.1.2.2 Exigencias de Calidad y Satisfacción del Cliente

La experiencia de usuario y la confiabilidad de las aplicaciones se han convertido en factores clave de competitividad. Las fallas en producción pueden afectar seriamente la satisfacción del cliente y la reputación de la marca.

Las pruebas automatizadas aseguran que cada versión del software cumple con los estándares de calidad y rendimiento, reduciendo la posibilidad de errores en producción.

Esto contribuye a una mejor experiencia de usuario y aumenta la confianza del cliente en el producto.

#### 3.1.2.3 Reducción de Costes a Largo Plazo

Si bien la implementación de una infraestructura de pruebas automatizadas requiere una inversión inicial, el ahorro en costes es significativo a largo plazo. Los tests automatizados pueden ejecutarse con alta frecuencia y en menor tiempo, reduciendo los costes de pruebas manuales repetitivas.

Las empresas evitan costos asociados a errores en producción, como pérdidas de ingresos, soporte técnico o ajustes de emergencia. La automatización permite detectar y corregir errores temprano en el ciclo de desarrollo, cuando su resolución es menos costosa.

#### 3.1.2.4. Competencia y Adopción por Parte de los Líderes del Mercado

Las empresas tecnológicas líderes (como Google, Amazon, Facebook) han adoptado ampliamente la automatización de pruebas, estableciendo un estándar en la industria. Esto ejerce presión sobre las empresas más pequeñas y en crecimiento para implementar prácticas de calidad similares si desean competir.

La alta adopción de herramientas de automatización de pruebas (como Selenium, Playwright, Cypress) y el crecimiento en el mercado de testing como servicio (TaaS) subrayan el valor que la automatización aporta en términos de calidad y eficiencia.

Fuentes: [mtheran](https://mteheran.dev/comparativa-entre-herramientas-de-automatizacion-playwright-selenium-y-cypress/), [clickup](https://clickup.com/es-ES/blog/228197/herramientas-de-automatizacion-de-pruebas)

### 3.1.3. Objetivos

**Objetivo general:** Demostrar la eficacia y el alcance del framework de automatización de pruebas en diferentes entornos de software, destacando su aplicación tanto en un entorno controlado de desarrollo/integración como en un entorno de producción robusto, como el de la universidad. Este objetivo se enfoca en validar cómo la planificación y ejecución de pruebas automatizadas contribuye a la calidad, seguridad y robustez de las aplicaciones.

**Objetivos específicos:**

1. **Desarrollar un entorno de prueba en Python:** Crear un aplicativo web propio con funcionalidades variadas que permitan implementar y validar diferentes tipos de pruebas automatizadas. Aunque las funcionalidades no estén necesariamente relacionadas en un entorno realista, servirán para demostrar el potencial del framework y sus capacidades para detectar errores y validar funcionalidades.
2. **Ejecutar pruebas en un entorno de producción:** Realizar una serie de pruebas automatizadas sobre aplicaciones de la universidad en un entorno de producción, gestionando y mitigando las limitaciones de seguridad de dicho entorno para evitar que los scripts sean bloqueados o identificados como ataques automatizados.
3. **Demostrar la adaptabilidad del framework en diferentes entornos:** Probar la flexibilidad y adaptabilidad del framework al operar tanto en un entorno de desarrollo controlado como en un entorno de producción robusto, demostrando su utilidad en distintos contextos de aplicación y niveles de seguridad.
4. **Mostrar los beneficios de una planificación y toma de requisitos eficaz:** Evaluar cómo una planificación adecuada de las pruebas y una toma de requisitos clara contribuyen al éxito de las pruebas automatizadas, ayudando a optimizar la ejecución y asegurando que las pruebas cumplan con los requisitos del sistema.

3.1.4. Requisitos técnicos

1. **Entorno de Desarrollo:**

Sistema Operativo: Windows, macOS o Linux.

Editor de Código: Se recomienda utilizar [Visual Studio Code](https://code.visualstudio.com/) para una mejor integración con las herramientas de desarrollo.

* 1. **Lenguaje de Programación:**

Python: Versión 3.8 o superior. Puede descargarse desde [python.org](https://www.python.org/downloads/).

* 1. **Gestión de Paquetes:**

pip: Gestor de paquetes de Python, generalmente incluido con la instalación de Python.

* 1. **Entorno Virtual:**

Se recomienda crear un entorno virtual para aislar las dependencias del proyecto. Puede crearse utilizando venv o herramientas similares.

* 1. **Dependencias del Proyecto:**

Instalar las dependencias listadas en el archivo requirements.txt del repositorio. Esto puede hacerse ejecutando:

bash

pip install -r requirements.txt

* 1. **Control de Versiones:**

**Git:** Para clonar el repositorio y gestionar el código fuente. Disponible en [git-scm.com](https://git-scm.com/).

* 1. **Configuración de Credenciales:**

Es necesario configurar las credenciales de acceso en el archivo config.values.json ubicado en src/helper/util/settings/. Este archivo debe contener las credenciales de usuario y contraseñas necesarias para las pruebas.

* 1. **Ejecución de Pruebas:**

Las pruebas pueden ejecutarse utilizando los comandos proporcionados en los archivos README.md, en función de ciertas etiquetas bien diferenciadas se podrán lanzar distintas baterías de tests

* 1. **Navegadores de playwright**

Se han de instalar los navegadores propios de playwright, en el propio proceso de instalación de playwright se indica como hacerlo.

1. **"Containerization":**

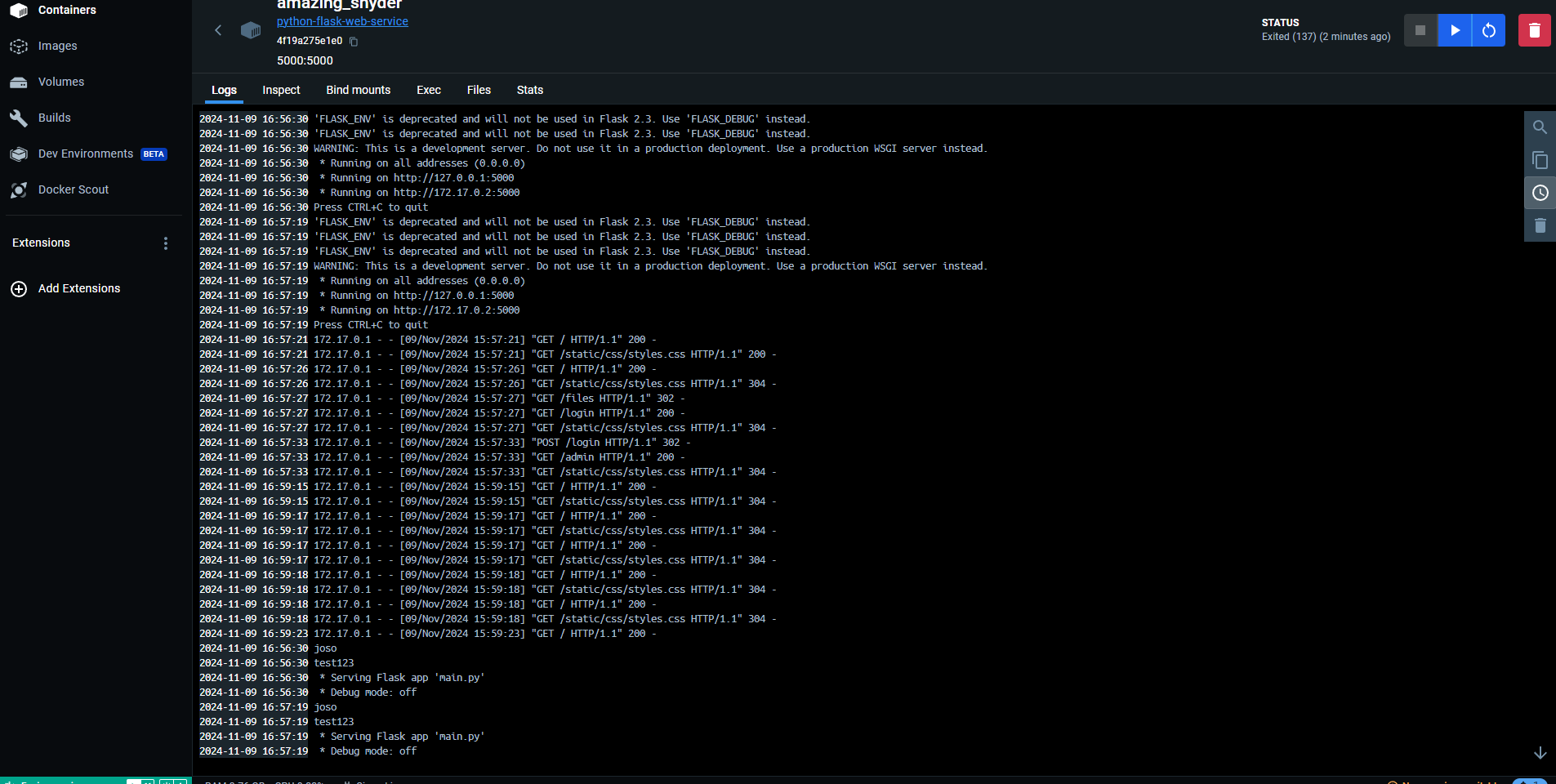
Por otro lado, se ha implementado el despliegue de todo el aplicativo en un contenedor Docker (solo la parte del desarrollo del aplicativo Python a testear).

A este proceso se le llama contenainerization o contenedorización en castellano (Siendo un anglicismo).

La motivación que tiene todo este proceso es la portabilidad del sistema, encapsulamiento de dependencias y, en general, facilitar la replicación de un aplicativo completo en otras máquinas.

Esto se hace a través de un contenedor docker, el cual es una abstracción a nivel de sistema operativo, como si fuera una máquina virtual, donde simplemente se ejecuta un programa junto a sus dependencias, emulando un equipo con un sistema operativo dado, pero, empleando recursos de la máquina anfitriona. Esto es una gran ventaja ya que la única dependencia externa que un equipo ha de tener para correr este mismo programa ha de ser Docker, o un sistema de contenedores similar

(e.g. kubernetes)

Captura de pantalla de un celular

Descripción generada automáticamente

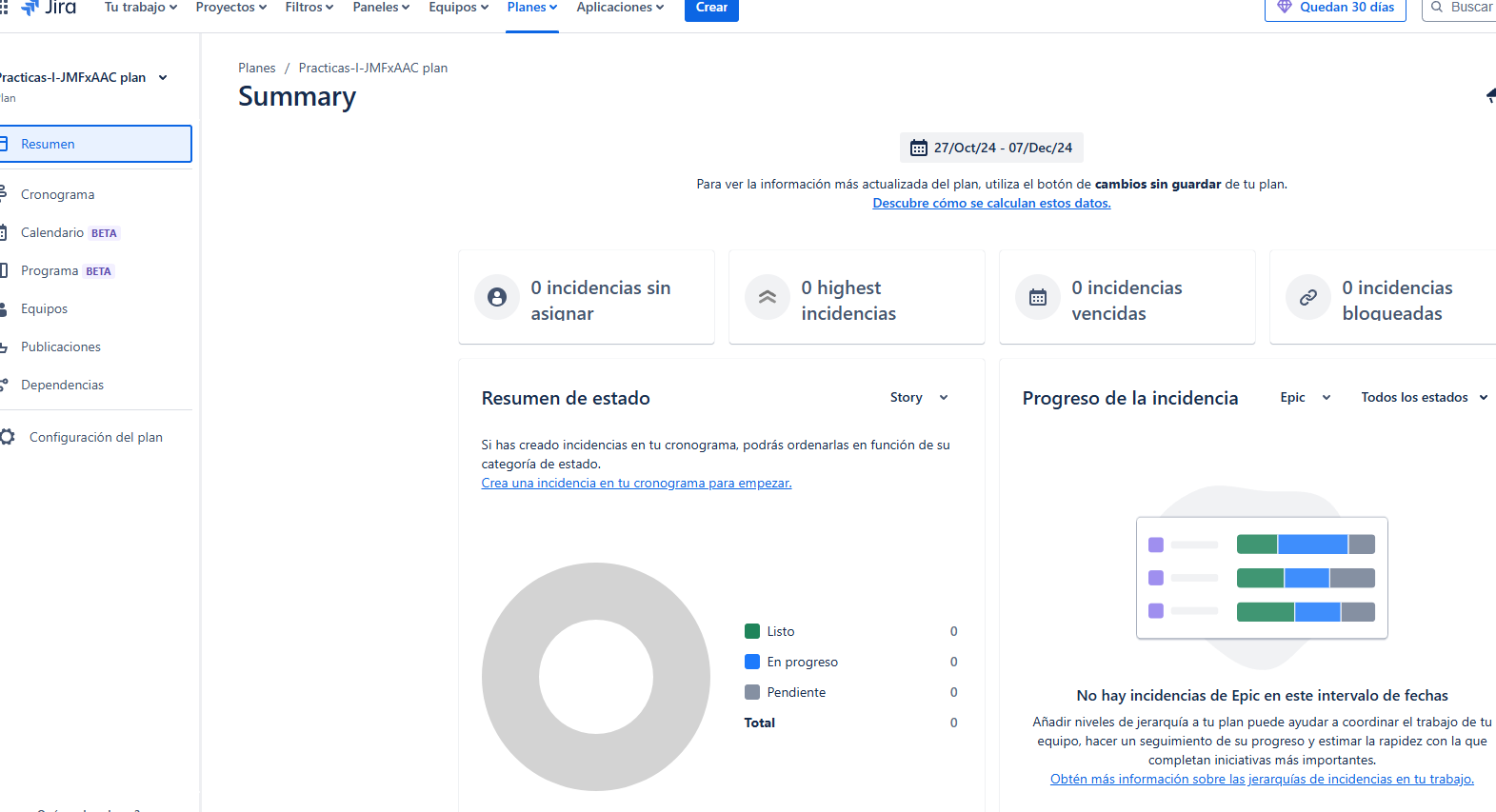
1. **Documentación de requisitos y gestión del proyecto**

Se han buscado alternativas para los aplicativos de la suite de Atlassian(Confluence y Jira), en un primer momento se han creado un par de instancias del proyecto con la versión gratuita de estos, se adjuntarán capturas de pantalla a continuación, por motivos de demostración, por otro lado, y de cara al momento en el que las licencias gratuitas de estos aplicativos finalicen, se han encontrado un par de alternativas

para la gestión del proyecto y la toma de requisitos.

Jira:

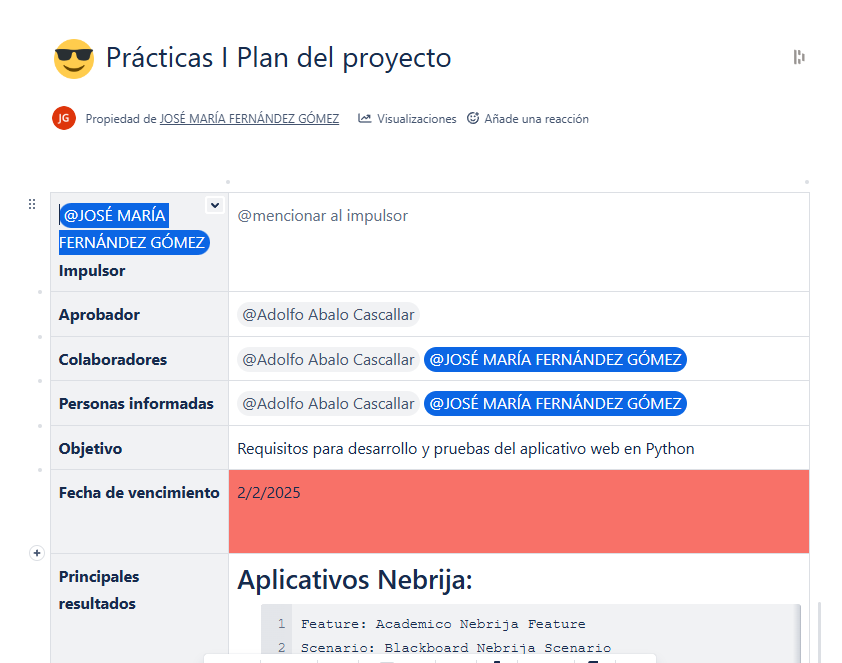
**Jira** (Atlassian®) es una herramienta de gestión de proyectos y seguimiento de tareas ampliamente utilizada por equipos de desarrollo de software y también por otros tipos de equipos para organizar y rastrear el progreso de proyectos, problemas y flujos de trabajo. Jira se destaca especialmente en la metodología ágil, proporcionando herramientas para el desarrollo iterativo y la entrega continua.

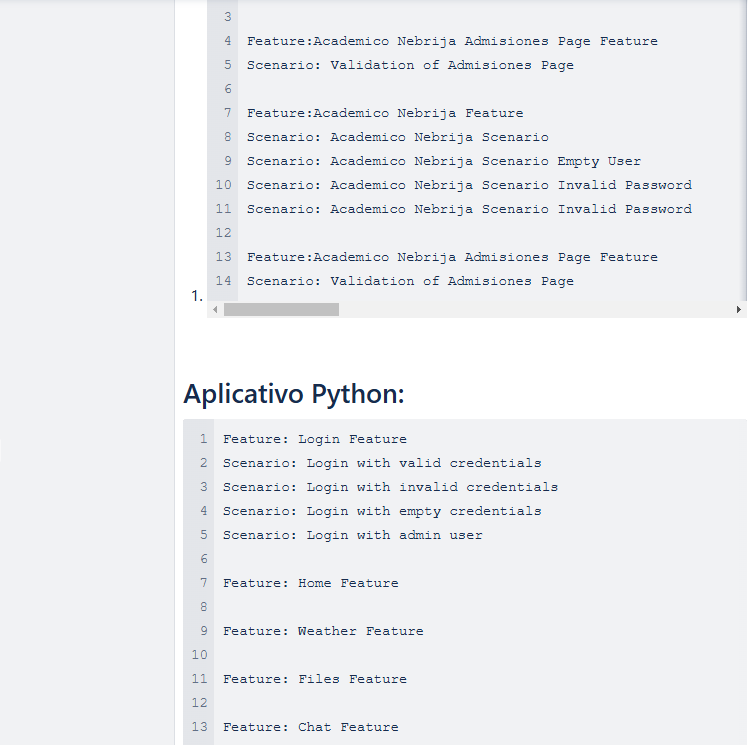


Confluence:

Confluence(Atlassian®) es un aplicativo de gestión de proyectos orientado a la definición y toma de requisitos para diferentes productos. Tiene diversas aplicaciones como pueden ser:

* **Gestión de proyectos**: Documentar avances, actas de reuniones y listas de tareas.
* **Base de conocimiento**: Crear una wiki corporativa o una biblioteca de procedimientos, manuales y recursos.
* **Documentación técnica**: Centralizar especificaciones de software, guías de instalación, y otros documentos técnicos.
* **Colaboración interdepartamental**: Facilitar la comunicación y la documentación entre equipos.





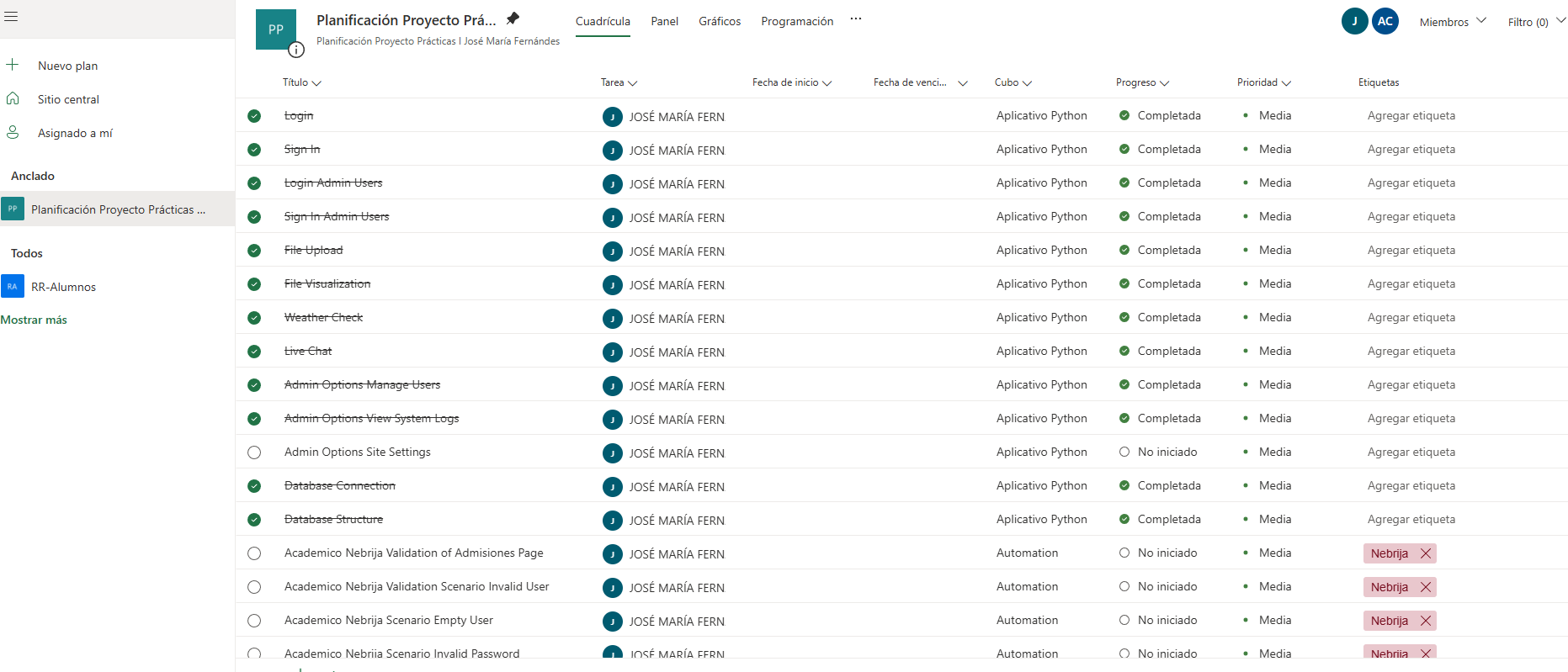
**3. Bis. Documentación de requisitos y gestión del proyecto**

Al ser Jira y Confluence aplicaciones que requieren de una licencia de pago, y en particular de una cuantiosa suma de dinero, se han buscado alternativas a estas para que, una vez finalizado el tiempo de prueba gratuita, se pueda seguir trabajando de una manera similar.

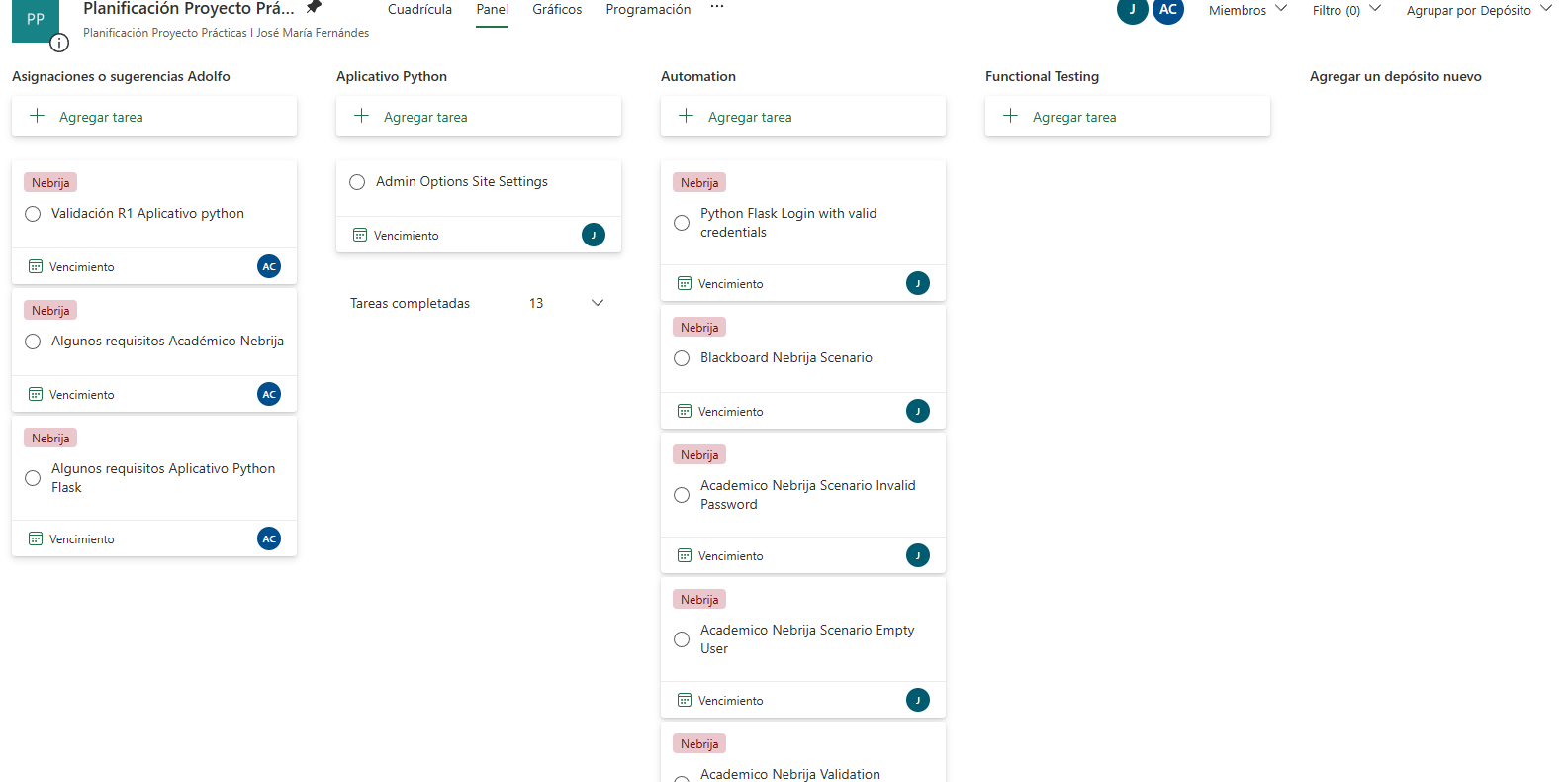
Las alternativas propuestas son:

Microsoft Planner para sustituir a Jira, ya que es un buen sistema, del que la universidad posee licencias, de cara a la organización, asignación y gestión de asignaciones de tareas o historias de usuario.

Link: [Microsoft Planner](https://planner.cloud.microsoft/Home/Planner/#/plangrid?groupId=e171cc2e-1d0f-4573-bb1d-ef5602f1fd50&planId=U3eIIiwgLEaTD5iZaryyYZYAHRdB)



Vista de To-do list



Vista de Scrum board

3.1.5. Análisis de mercado

3.2. Marco técnico

3.3. Equipo de trabajo y metodología

3.4. Proyecto

3.4.1. Resumen de contribuciones y productos desarrollados

##### Aplicativo de Pruebas en Entorno de Desarrollo:

* 1. Creación de un aplicativo web en Python, diseñado específicamente para pruebas, con múltiples funcionalidades que sirven para demostrar el alcance y capacidades del framework de automatización.
  2. Este entorno permite al alumno realizar y validar pruebas automatizadas en un entorno controlado, mostrando ejemplos de pruebas funcionales y de integración sobre distintas funcionalidades de la aplicación, simulando lo que podría ser un aplicativo real.

##### Scripts de Pruebas Automatizadas para el Entorno de Producción:

* 1. Desarrollo de una batería de tests automatizados aplicados sobre aplicaciones de la universidad, demostrando el uso de pruebas en un entorno de producción.

##### Scripts de Ejecución para Diferentes Casos de Prueba:

* 1. Desarrollo de scripts específicos para distintos escenarios, como pruebas de autenticación, acceso a páginas específicas y validación de formularios, entre otros.
  2. Estos scripts, creados en un entorno de desarrollo e integración, están orientados a cubrir casos de prueba relevantes para la funcionalidad y seguridad de los aplicativos.

### 

### 3.4.2. Planificación temporal

Imagen que contiene Diagrama

Descripción generada automáticamente

Escala de tiempo

Descripción generada automáticamente con confianza media

Escala de tiempo

Descripción generada automáticamente

Escala de tiempo

Descripción generada automáticamente

3.4.3. Recursos empleados

1. Gestor de proyectos y de toma de requisitos similar a Jira y Confluence

3.4.4. Trabajo desarrollado  
3.5. Resultados y discusión.  
3.6. Conclusiones.  
3.7. Líneas futuras.

4. BIBLIOGRAFÍA.

# Referencias

Deloitte. (s.f.). Obtenido de Src: https://www.deloitte.com/es/es/services/consulting/services/technology-strategy-and-transformation.html

Marketing Scoop. (7 de 6 de 2024). Obtenido de https://www.marketingscoop.com/tech/proxy/puppeteer-vs-playwright-vs-selenium-the-ultimate-comparison-for-web-scraping-and-browser-automation-in-2024/?utm\_source=chatgpt.com

5. ANEXOS