

Universidad ORT Uruguay

Facultad de Ingeniería

Obligatorio – Informe Final

Ingeniería de Software Ágil 2

Ana Laura Montes de Oca (146669)

Giuliano Rositto (256201)

Sebastián Silvera (242951)

Docentes:

Carina Fontan

Alvaro Ortas

Entregado como Informe Final de Obligatorio

<https://github.com/IngSoft-ISA2-2023-2/obligatorio-montes-de-oca-rositto-silvera>

13/Noviembre 2023

DECLARACIÓN DE AUTORÍA

Nosotros, Ana Laura Montes de Oca (146669), Giuliano Rositto (256201) y Sebastián Silvera (242951) declaramos que el trabajo que se presenta en esa obra es de nuestra propia mano. Podemos asegurar que:

- La obra fue producida en su totalidad mientras realizábamos el curso de Ingeniería de Software Ágil 2
- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad.
- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra
- En la obra, hemos acusado recibo de las ayudas recibidas
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente qué fue contribuido por otros, y qué fue contribuido por nosotros
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

Contenido

1.	Resumen de Gestión del Proyecto	4
2.	Reflexiones sobre el aprendizaje.....	8
3.	Lecciones aprendidas.	11
4.	Conclusiones.	12
5.	Guía de instalación para desarrollo y despliegue en producción.....	14

1. Resumen de Gestión del Proyecto

Durante la gestión del proyecto, se llevaron a cabo diversas actividades con el objetivo de mejorar y ofrecer mantenimiento a una aplicación ya existente. El enfoque principal ha sido incrementar el valor entregado a los usuarios, utilizando una metodología basada en Kanban y siguiendo un ciclo de vida incremental del producto.

Cada opción del índice de este informe se linkea con el ReadMe del repositorio pudiendo acceder a cada entrega con sus respectivos entregables y/o archivos necesarios.

Entrega 1

Pasos y Actividades Realizadas Entrega 1:

1. **Planificación y Adopción de Metodología:**
 - Se adoptó la metodología Kanban (en integración con SCRUM, algunos artefactos) para gestionar el flujo de trabajo y optimizar la entrega de productos y servicios.
 - Se creó un proyecto en un repositorio, estableciendo tableros Kanban dinámicos para gestionar las entregas.
2. **Ingeniería de Requerimientos:**
 - Se aplicó un proceso sistemático para el descubrimiento, desarrollo, seguimiento, análisis, clasificación, comunicación y gestión de requerimientos.
 - La ingeniería de requerimientos se integró en el proceso Planning-Do-Check, adaptándola al enfoque incremental.
3. **Adaptación de Principios Kanban:**
 - Se visualizó el flujo de trabajo mediante tableros Kanban y se limitó el trabajo en curso (WIP).
 - Se implementaron ciclos de feedback a través de retrospectivas y stand-ups.
 - Se explicitaron políticas y procedimientos, y se fomentó la mejora continua mediante la colaboración.
4. **Control de Versiones:**
 - Se estableció un repositorio en GitHub utilizando inicialmente la metodología GitFlow y con la intención de migrar gradualmente a Trunk-Based Development.
 - Se crearon ramas paralelas desde la rama principal (main) y se trabajó en ellas, realizando fusiones según el flujo de trabajo.

Productos Entregados:

- Se creó un proyecto en GitHub con una estructura de ramas y se realizaron entregas incrementales.
- Se generaron informes de avance detallando issues identificadas y acciones correctivas.
- Resultados de Clockyfi.
- Informe final del trabajo realizado (en ReadME)

Entrega 2:

- **Actividades Realizadas:**

- Creación del proyecto "Entrega 2" con un tablero más detallado (sustentable)
- Incorporación de columnas para pruebas unitarias, calidad del código y métricas.
- Inclusión de Fechas a la hora de reportar para clasificación y ordenamiento.
- Desglose de tareas complejas y asignación con visibilidad del responsable.
- Cambios en el Definition of Done la cual pasa a incluir Validación y Testing.
- Mantenimiento de la regla de un desarrollador por tarea.
- Asignación de tareas con visibilidad del responsable.
- StandUps frecuentes para feedback útil.
- Limitaciones de tiempo para instancias de StandUp, Review y Retrospectiva.
- Aprendizaje y experimentación con GitHub Actions para pipeline automático.
- Registro mejorado en Clockify, mapeando con columnas del tablero.
- Transición a desarrollo Trunk-Based con rama principal "main".
- Creación de ramas paralelas limitadas a 2 días.
- Funciones automatizadas para verificación de build, instalación de dependencias y pruebas unitarias.
-

Productos Entregados:

- Tablero Kanban mejorado con mayor detalle y visibilidad.
- Mejora en la gestión del flujo y limitación del trabajo en progreso.
- Reporte detallado sobre cambios en las políticas.
- Utilización de Github Actions para automatización. Si bien no es un producto entregado como tal si lo consideramos una parte relevante de nuestro producto en GIT.
- Entrega del código con las bugs seleccionados, arreglados por propio trabajo.
- Herramientas de automatización ubicadas en las columnas de Testeo y Validación.

Entrega 3

Actividades Realizadas:

- Creación del proyecto "Entrega 3" para la nueva definición del tablero Kanban.
- Establecimiento de fechas clave (Start, End, Front, Back) para controlar el flujo.
- Desarrollo y definición de requerimientos en GitHub con el formato 'Como... Quiero... Para...'.

- Implementación del Front End y Back End con pruebas automatizadas utilizando SpecFlow.
- Mejora del código mediante refactoring.
- Pruebas de integración con PostMan y la propia aplicación.
- Mejora en el proceso de merge sobre main y menor uso de ramas paralelas.
- Identificación de historias de usuario y desarrollo estructural Frontend.
- Implementación de SpecFlow y metodología BDD para Backend.
- Conexión y verificación del sistema producido.
- Limitación del pipeline a funcionalidades separadas para cada miembro del equipo.

Productos Entregados:

- Tablero Kanban actualizado.
- Requerimientos definidos y documentados.
- Desarrollo completo del Front End y Back End con pruebas automatizadas.
- Versionado mejorado y estructura de código optimizada.
- Implementación de SpecFlow y BDD en el pipeline.

Entrega 4:

Actividades Realizadas:

- Apropiación de Herramientas y Entendimiento de Requerimientos:
- Mejora en el Registro de Clockify:
- Unificación de la Herramienta para la Segunda Entrega:
- Mejora de Métricas en Clockify:
 - Se mejoraron los registros en Clockify, permitiendo obtener tareas agrupadas por etiquetas correspondientes al tablero, facilitando el análisis del CycleTime y la identificación de posibles demoras.
- Transferencia de Tareas No Completadas:
 - Algunas tareas de la entrega 3, como la integración de BDD, no se completaron en fecha y se transfirieron a la entrega 4.

Productos Entregados o Entregables:

- **Archivo Excel - Entrega 4:**
 - Incluye un resumen de cada etapa, con detalles sobre CycleTime, LeadTime, y FlowEfficiency.
- **Métricas y Conclusiones de las mismas**
- **Selenium:** se utilizó Selenium, realizando pruebas que evaluaron el funcionamiento de las funcionalidades implementadas en entregas anteriores y se entregó código de Selenium para dichas pruebas.

En todas las entregas se entregó un informe y link a las retrospective.

Issues:

Total de Issues reportadas: 32

Issues que son Bugs (severidad crítica o alta): 12, específicamente:

- Issue 24: Error en Test de Funcionalidad a nivel de código (Prioridad: Inmediata - Severidad: Crítica)
- Issue 20: Discrepancia en el número de artículos solicitados durante la compra como usuario anónimo (Prioridad: Alta - Severidad: Crítica)
- Issue 19: Falta de botón claro para volver atrás en la aplicación (Prioridad: Media - Severidad: Leve)
- Issue 17: Creación de solicitudes de stock negativas o valores cero (Prioridad: Media - Severidad: Crítica)
- Issue 16: Creación de Medicamento al lanzar una excepción esta es genérica (Prioridad: Media - Severidad: Leve)
- Issue 15: Cuando el cliente llama a un servidor offline no se le informa de esto (Prioridad: Baja - Severidad: Leve)
- Issue 14: Hora de Transacción desfasada (Prioridad: Baja - Severidad: Leve)
- Issue 13: Falta de sistema de Imágenes (Prioridad: Baja - Severidad: Leve)
- Issue 12: No se informa al Usuario cuando una farmacia no posee ningún medicamento que coincida con la búsqueda (Prioridad: Baja - Severidad: Leve)
- Issue 11: Búsqueda de remedios filtra por título exacto (Prioridad: Alta - Severidad: Mayor)
- Issue 3: La funcionalidad del Admin: Alta de Farmacia - Nombre duplicado (Prioridad: Inmediata - Severidad: Crítica)
- Issue 2: La funcionalidad del Admin: Alta de Farmacia bug (Prioridad: Inmediata - Severidad: Crítica)

Tareas a Desarrollar (excluyendo Bugs): 20, que incluyen diversas tareas de desarrollo y pruebas, como eliminación de productos, modificación de productos, alta de productos, exploración de funciones de dueño, entre otras.

Issues reportadas y desarrolladas: [Ver Anexo](#) (Anexo issues.xls)

2. Reflexiones sobre el aprendizaje.

• Aplicar un marco de gestión ágil.

Vimos que a lo largo de nuestro proyecto DevOps aplicamos los 2 marcos de gestión ágil integrados, por un lado: Scrum con la definición de roles y ceremonias (explicado en el ReadMe) y también Kanban con la definición de tableros y nuevo proceso de ingeniería, alineando los mismos durante el desarrollo de las diferentes entregas.

La definición de roles que proporciona SCRUM, nos ayudó a mejorar el proceso de gestión. Vimos los roles de scrum master (SM) y product owner como fundamentales además de los propios requeridos para el proyecto: desarrolladores y testers.

D: Drop: qué hicimos mal y debemos dejar de hacer?

No considerar un SM en principio hizo que nuestro trabajo requiriera orden y organización, por lo que lo vimos su necesidad y lo definimos rápidamente.

En el cálculo de métricas vimos que nuestro LeadTime era mayor cuando ingresábamos las issues lo antes posible, aún cuando no teníamos los conocimientos teóricos para empezar su desarrollo. Estos tiempos se hubieran mejorado si las issues se ingresaban lo más tarde posible al tablero, próximo a su desarrollo.

A: Add: qué no hicimos y deberíamos comenzar a hacer?

(A): Las ceremonias asíncronas no siempre lograban resultados positivos, fue necesario comenzar a organizarnos con reuniones sincrónicas para poder alinearlos, las standups sincrónicas permitieron entender las dificultades del momento para poder mejorar.

K: Keep, (¿Qué hicimos bien y deberíamos continuar haciendo?)

Continuar logrando compromiso y dedicación sobre todo durante el período de las mini-entregas. Vimos que hubo una evolución en cuanto a dedicación consistente en aquellos períodos en los que las entregas eran de 2 o 3 semanas. Esto fue debido a que la curva de aprendizaje fue mayor, por contenidos nuevos que se fueron adquiriendo.

El uso del tablero y su alineación con el proceso de ingeniería se fue refinando, pasando de un tablero simple de 3, 4 columnas a un tablero sustentable, donde identificamos cada issue tarea qué ciclo recorría.

La columna de Gestión siempre estuvo presente, agrupando las actividades como ceremonias, redacción de informes. Vimos que los registros de esfuerzo, para las métricas si bien se enfocaron en los de desarrollos de issues (bugs o nuevas funcionalidades) también tuvieron su magnitud para este tipo de actividades. Esto se reflejó en los informes de Clockify.

I: Improve (¿Qué hicimos relativamente bien, pero podríamos mejorar?).

Las retrospectivas son las ceremonias donde mejor se identifican y visualizan nuestras fortalezas y debilidades, pudiendo recoger feedback de nuestro proceso para poder mejorarlo. Se documentaron correctamente a través de video, evidencias en la plataforma metro-retro y conclusiones en ReadMe. Podemos mejorar como visualizar las acciones correctivas para poder tenerlas en cuenta en próximas iteraciones.

- **Analizar la deuda técnica.**

Las características del análisis de cada issue se trabajó en base a su descripción (breve reseña de en qué consiste el bug reportado), impacto (cual es el efecto para el negocio que conlleva no reparar este bug), solución ideal (para reparar el bug), plan de acción (para abordar su identificación, análisis y diseño de su solución para luego reparación) y clasificación (severidad y prioridad)

I:Entendemos, que la clasificación exhaustiva en la cual nos basamos nos permitió llegar a trabajar con la issues de la mejor manera posible.

Vemos como oportunidades de mejora seguir optimizando los tiempos en el desarrollo y testing de las issues como bugs y nuevas funcionalidades para lograr mayor calidad en lo desarrollado. La corrección de algunos bugs pudieron producir nuevos 'bugs' que se identificaron y se reportaron o simplemente no se identificaron por falta de testing, aquí priorizamos la completitud de las tareas solicitadas, pero todos los bugs fueron reportados en el repositorio como issues.

- **Implementar un repositorio y procedimientos de versionado.**

El versionado se fue optimizando, al comienzo, cada desarrollador manejaba sus ramas independientes que luego mergeaba a main. Luego pasamos a crear una branch Entrega nro. que se generaba como realease.

Mejoramos en cuanto al merge sobre main, los commit's fueron más frecuentes para la reparación de errores, e incluso algunos desarrolladores trabajamos con menos ramas paralelas, confiando más en el proceso de trunk based.

A: Para la integración de nuevas funcionalidades, notamos que aún falta afianzar el proceso, por lo que se continúan viendo ramas paralelas durante la entrega 3. Entendemos que falta tiempo y confianza en el proceso para poder implementar una nueva funcionalidad directamente en la rama main.

K: Notamos que el proceso es más ágil, logramos la integración y despliegue continuo cuanto más confiamos en él, por lo que sería bueno seguir practicándolo.

I: Sin duda para mejorar la confianza en el proceso para poder lograr la integración continua. También podemos mejorar nuestro LeadTime ya que los tiempos de despliegues frecuentes bajarían.

- **Crear un pipeline con eventos y acciones.**

El pipeline fue creado para la entrega 2 y no recibió cambios en entregas posteriores, en este implementamos un sistema que revisa que la aplicación buildee correctamente en ambos ambientes y revisa si las pruebas de backend corren correctamente.

En un inicio fue útil para asegurarnos que momentos de integración complejos como los merges de distintos trabajos hayan sido exitosos.

Mas adelante en el desarrollo una de las pruebas que aparecían previamente y que no supimos analizar o resolver fallaba y ante esto también lo hacia el pipeline, debido a que no

resolvimos este problema dejamos de utilizar también la utilidad implementada por el pipeline

A futuras iteraciones consideramos que deberíamos enfocarnos en resolver el problema que bloquea el pipeline para poder usar mejor su funcionalidad y a su vez ver la posibilidad de automatizar distintos procesos extras.

Generar escenarios de testing desde la perspectiva del usuario.

Sin duda el cumplir con este objetivo es el que más oportunidades de mejora tiene. La incorporación de BDD como técnica para desarrollo de escenarios nos implicó mayores esfuerzos en horas y no logramos incorporar totalmente la técnica por falta de tiempo y por falta de tiempos.

D: No logramos comprender la herramienta SpecFlow en su totalidad, nos demandó mayor tiempo su comprensión, por lo que optamos por completar la funcionalidad realizando test funcional para su completitud del DONE. Necesitamos mayores tiempos para la adquisición de la herramienta e integración con nuestro ide de trabajo.

A: Agregar más escenarios y test en todo el ciclo de BDD

K: Seguir investigando la herramienta en próximas iteraciones. Notamos que quizá no se acopló a nuestro ide seleccionado, ya que a veces dejaban de funcionar los test anteriores que ya teníamos.

I: BDD es una técnica que sin duda favorece la calidad de nuestro software acercando al cliente un mejor entendimiento en las necesidades del negocio, seguir mejorando en el conocimiento para lograr la integración completa a través de la incorporación de más y mejores escenarios.

Reflexionar sobre DevOps.

Podemos reflexionar sobre DevOps como proceso o conjunto de técnicas que buscan mejorar la colaboración mutua entre equipos de Dev y Ops acelerando el ciclo de vida del desarrollo de software desde la planificación (PLAN), desarrollos (DO) hasta la implementación y operación con un enfoque de automatización y mejora continua (CHECK), proporcionando buenas prácticas durante todo el proceso acompañados de CI/CD.

Integrar prácticas de QA en el pipeline y gestionar el feedback.

Si bien se implementó la guía de instalación de deploy como parte de la Entrega 5) no tuvimos oportunidad de aplicar QA como parte del proceso para gestionar la Calidad y Feedack.

3. Lecciones aprendidas.

Lección aprendida #1:

Sobre los roles:

Si necesitamos gestionar un proyecto dentro del marco ágil para cumplir con las reviews sugerimos identificar y definir un único Product Owner (PO).

Anécdota: al comienzo del desarrollo de nuestro proyecto decidimos que los 3 desarrolladores seríamos PO, con la intención de fomentar el espíritu crítico y fomentar discusiones constructivas. Observamos que no logramos un nivel de criticidad similar a un PO del lado del cliente.

Lección aprendida #2:

Sobre SpecFlow

Si queremos utilizar la herramienta SpecFlow para cumplir con el proceso de BDD sugerimos investigar la integración con el IDE a utilizar y también la herramienta en sí a través de tutoriales y reservando tiempo para la adquisición del aprendizaje de esta herramienta.

Anécdota: Nos pasó en esta integración que logramos integrar un escenario y al intentar agregar otro escenario al test anterior había que hacerlo como una feature nueva y no como un escenario más dentro de la feature. La herramienta no nos resultó amigable para poder realizar la integración de manera fluida.

Lección aprendida #3:

Sobre Flujo de valor-LeadTime

Si queremos optimizar el flujo de valor, mejorando el LeadTime podemos hacerlo controlando cuando ingresamos la issue en el TODO.

Anécdota: Dentro de una de las entregas para optimizar los tiempos agregamos todas las issues al Tablero para organizarnos y ahorrar tiempo luego en asignar a los desarrolladores, ya que cada uno podría ir analizando cuál issue tomaría. Esta técnica, más allá de favorecernos nos aumentó el LeadTime.

Lección aprendida #4:

Sobre Selenium

La utilización de Selenium fue una herramienta invaluable que nos permitió ejecutar pruebas de forma automática, acelerando significativamente el proceso y eliminando la dependencia del personal humano para la ejecución de las pruebas, lo que aumentó la velocidad y la certeza en los resultados. Aunque la curva de aprendizaje fue empinada al principio, la capacidad de determinar identificadores únicos en nuestro front-end simplificó y agilizó las pruebas.

En resumen, Selenium resultó ser una herramienta desconocida para nuestro equipo, pero su capacidad para detectar errores de manera temprana lo convirtió en una adición altamente beneficiosa a nuestro conjunto de herramientas.

Anécdota: Durante las pruebas iniciales encontramos dificultades para identificar componentes del Front, utilizando id's únicos logramos detectar los componentes más fácilmente.

4. Conclusiones.

Las conclusiones del informe deberían poder responder a las siguientes interrogantes:

- **¿Qué significan los resultados de lo investigado, aplicado y/o solucionado?**

- 1) Podemos concluir que los bugs corregidos y features implementadas aportaron a reducir la deuda técnica inicial identificada. Esta reducción fue acompañada del ejercicio de buenas prácticas y adquisición de nuevas herramientas para el equipo.
- 2) [Evolución de las métricas:](#)

FlowEfficiency:

Entrega 1 < Entrega 2 < Entrega 3 < Entrega 4

8% < 39% < 62% < 73%

Podemos concluir que nuestro proceso fue mejorando a medida que avanzaba, aunque las tareas que se ingresaron en el tablero como features y bugs demandaron mayor esfuerzo. También destacamos que el poco tiempo entre las entregas hizo que no pudiéramos testear con total dedicación al 100% algunas funcionalidades pudiendo introducirse nuevos bugs. Como lo mencionamos en la retrospectiva, tuvimos algunos tiempos de espera que hicieron aumentar el LeadTime, sobre todo en la 4ta. entrega. Esto se debió a que se crearon las issues con tiempo de antelación antes de que pudiéramos empezar su desarrollo, sobre todo porque no se habían completado los temas en el dictado.

- **¿Qué consecuencias/implicancias tiene lo anterior?**

Beneficiosas: logramos introducir un marco de gestión ágil para acompañar el proceso de DevOps.

Oportunidades de mejora: vemos que en el desarrollo de las diferentes entregas pudimos habernos enfocado en la solución de algunos errores como la falla en el pipeline que nos hubieran permitido completar ese proceso.

- **¿Por qué son importantes esas implicancias o consecuencias?**

El reducir la deuda técnica redundó en un software de mejor calidad y aplicando las buenas prácticas de DevOps como la integración y deploy continuo pudimos incorporar la metodología ágil acelerando los procesos.

Al no completar los procesos, transfiriendo la falla, dejamos de prestar atención a la herramienta y no notamos si hubo fallas nuevas.

- **¿A dónde nos conducen?**

Durante el proyecto mejoramos el proceso de ingeniería. Esto nos permite minimizar los tiempos de desarrollo y puesta en producción. En contrapartida, en la medida de no eliminar los errores, los beneficios brindados por las herramientas se vuelven menos eficientes.

5. Guía de instalación para desarrollo y despliegue en producción.

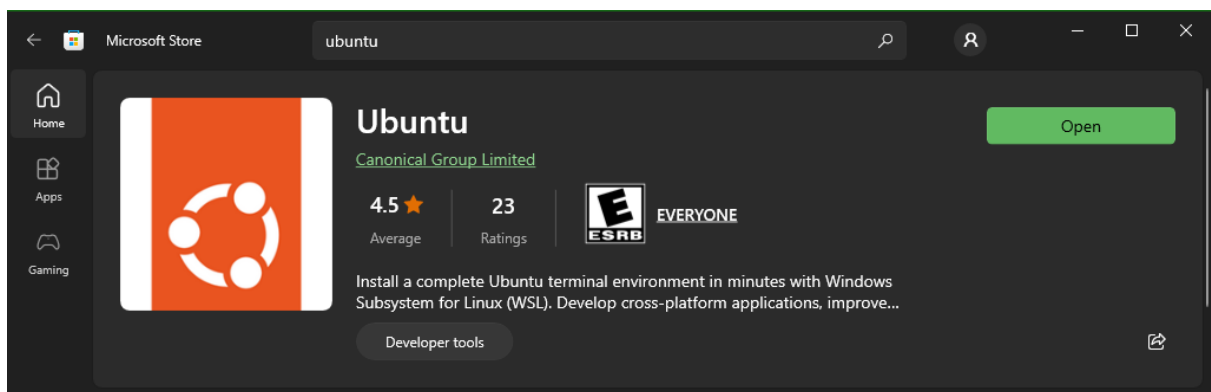
Deploy FrontEnd

Descargar las carpetas del repositorio:

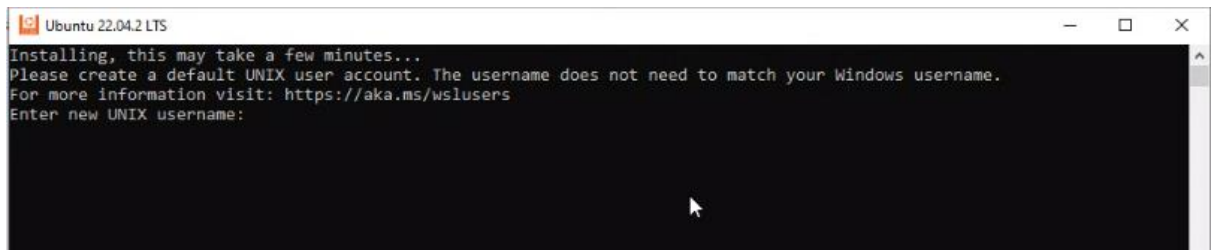
Haremos uso de la carpeta `pharma-go` ubicada en `${Nombre-repositorio}/Codigo/Frontend/dist/`

Nos separamos esta carpeta y empezaremos la instalación de nuestro sistema:

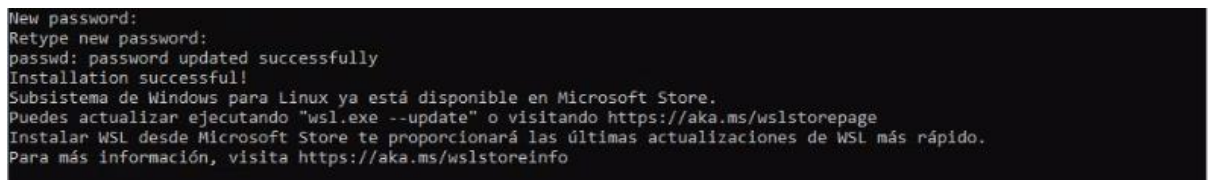
Usaremos Ubuntu Desktop para Windows para deployar nuestro servidor, este es accesible a través de la tienda de microsoft



Una vez instalada abrimos la aplicación haciendo click en Open



Esperamos a que finalice la instalación, ingresamos un nombre de usuario y una contraseña



Una vez ubuntu está preparado para ser usado prepararemos las herramientas a utilizar:

```
sudo apt update && sudo apt upgrade
```

```
sudo apt install openssh-server
```

```
sudo systemctl enable --now ssh
```

```
sudo apt-get install apache2
```

```
sudo apt-get install ufw
```

```
sudo ufw allow http
```

```
sudo ufw allow https
```

```
sudo ufw allow Apache
```

```
sudo ufw allow 22
```

obtengo la dirección ip de mi ubuntu Desktop

```
hostname -I
```

transferimos nuestra carpeta pharma-go a ubuntu:

para eso usaremos command prompt haciendo Win+R y escribiendo cmd

Con cmd abierto ejecutamos el siguiente comando reemplazando los campos \${}

```
scp -r ${camino/a/pharma-go} ${nombreUsuarioUbuntu}@${direccion ip conseguida}:
```

luego se nos solicita la contraseña del usuario de ubuntu ingresado, la ingresamos y finalizamos la transferencia

ej:

```
C:\Windows\system32>scp -r C:\Users\giuli\Escritorio\pharma-go giulimax@172.29.59.113:
giulimax@172.29.59.113's password:
```

Nuevamente en Ubuntu seguimos ejecutando los siguientes comandos:

```
cd ~
```

```
sudo mv pharma-go /var/www
```

```
sudo nano /etc/apache2/sites-available/pharma-go.conf
```

Escribimos lo siguiente:

```
<VirtualHost *:80>
```

```
ServerAdmin owner@pharma-go.com
```

```
ServerName pharma-go.com
```

```
ServerAlias www.pharma-go.com
```

```
DocumentRoot /var/www/pharma-go
```

```
ErrorLog ${APACHE_LOG_DIR}/error.log
```

```
CustomLog ${APACHE_LOG_DIR}/access.log combined
```

```
<Directory /var/www/pharma-go>
```

```
FallbackResource /index.html
```

```
AllowOverride All
```

```
Require all granted
```

```
</Directory>
```

```
</VirtualHost>
```

Hacemos Ctrl+O, Enter y Ctrl+X

Seguimos ejecutando:

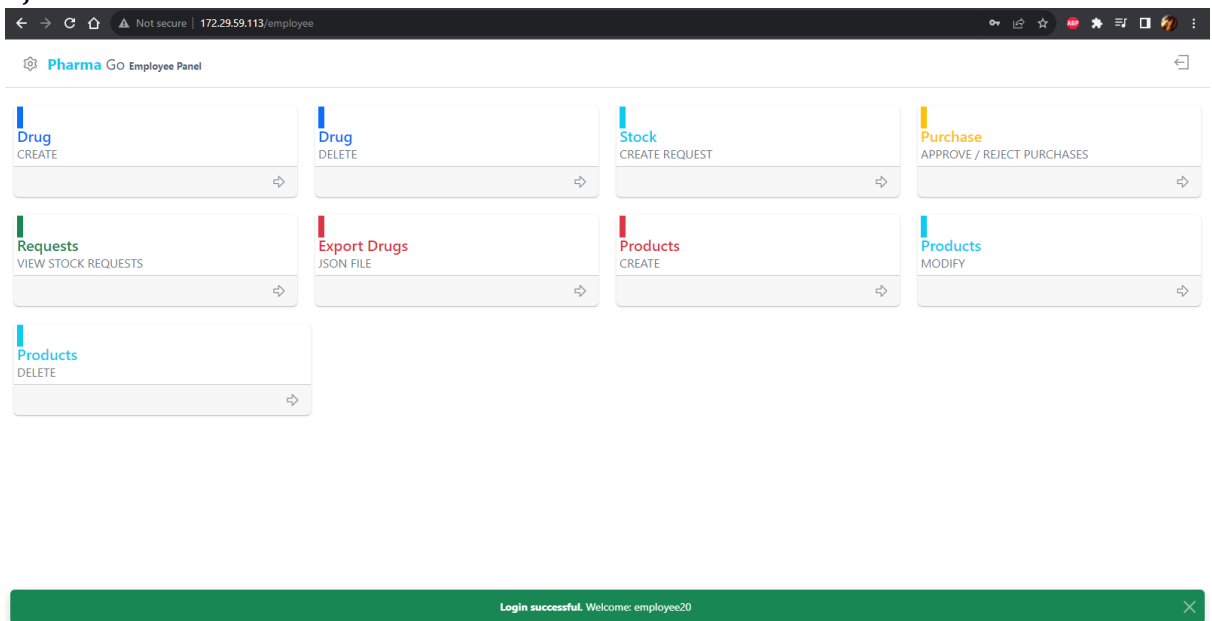
```
sudo a2dissite 000-default.conf
```

```
sudo a2ensite pharma-go
```

```
sudo systemctl reload apache2
```


Finalmente si accedemos a nuestro navegador de preferencia en nuestra máquina y escribimos la dirección IP previamente consultada accederemos a nuestro sitio

Ej:

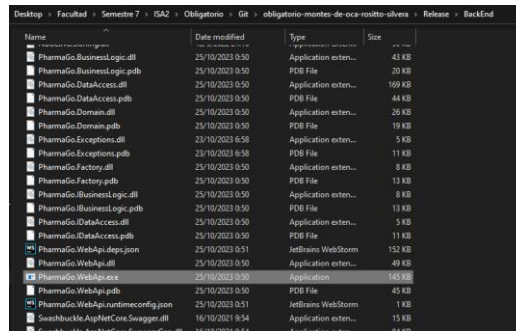


La ip es de carácter temporal, cada vez que reiniciemos nuestra instancia deberemos consultar nuevamente la ip actual con `hostname -i` y usar esa dirección en el buscador

Deploy Backend


Para hacer deploy del Backend descargamos del repositorio la carpeta Release/Backend, en esta carpeta encontraremos varios archivos y entre ellos el ejecutable PharmaGo.WebApi.exe.

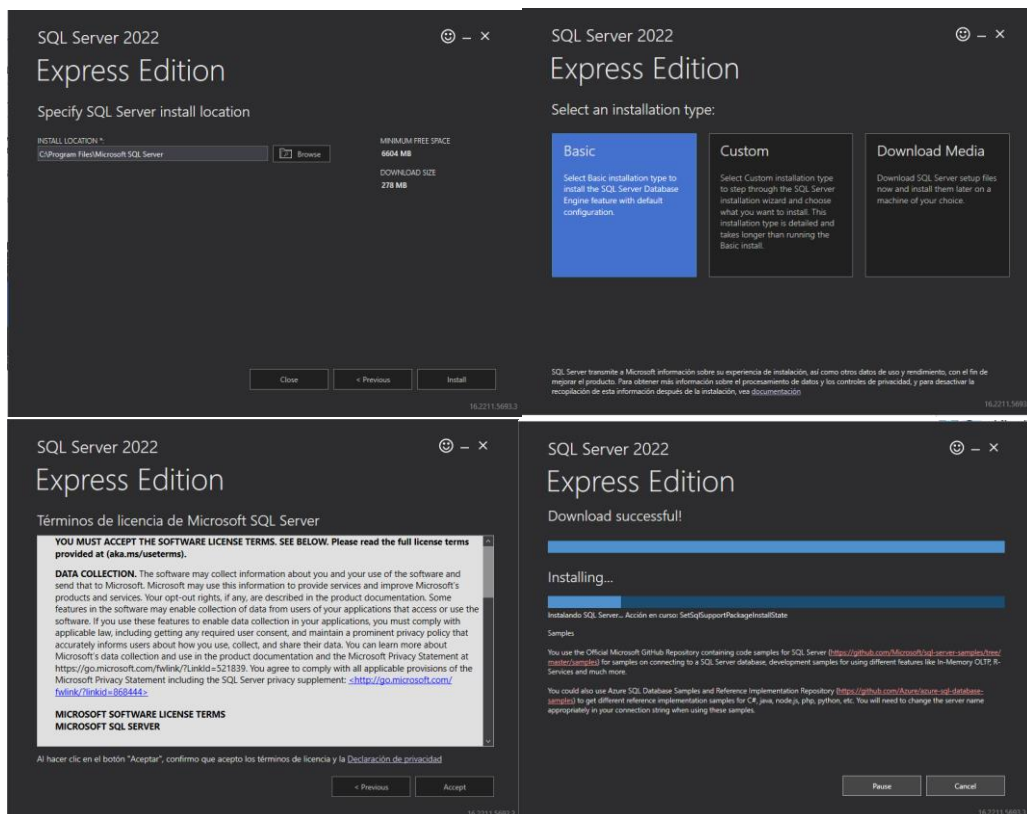
Movemos la carpeta Backend a la ubicación que querramos y ejecutaremos PharmaGo.WebApi.exe cuando queramos iniciar nuestro Backend.



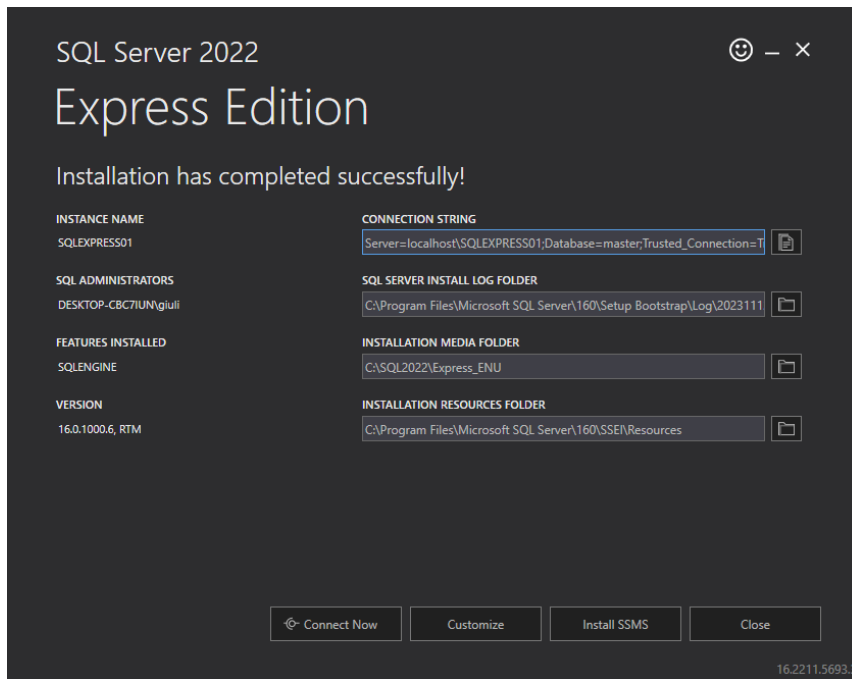
Precisamos instalar nuestra base de datos para habilitar la funcionalidad, para esto instalaremos Microsoft SQL Server Express 2022 usando el siguiente link

<https://www.microsoft.com/es-es/download/details.aspx?id=104781>

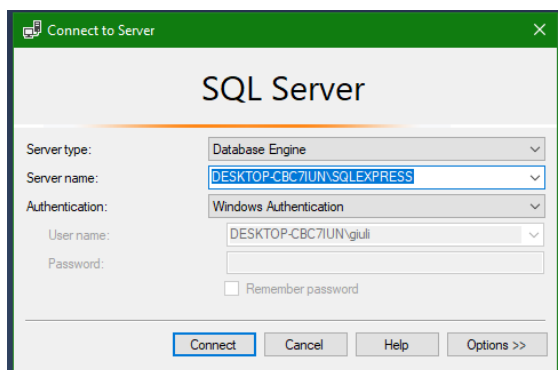
Abrimos el ejecutable , elegimos la opcion basic, aceptamos los terminos de licencia y elegimos nuestra ubicacion de instalacion y apretamos install



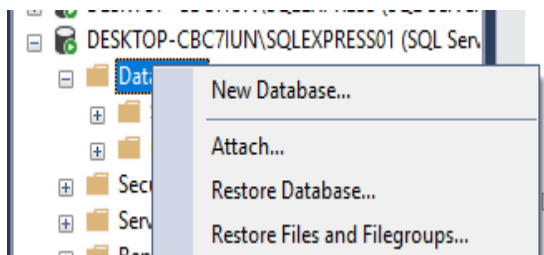
Una vez finalizada la instalación, registramos los datos que nos muestran y si no tenemos la herramienta instalada hacemos click en install SSMS:



Abrimos SSMS y en el pop up que nos aparece indicamos el nombre de nuestra instancia y hacemos click en connect



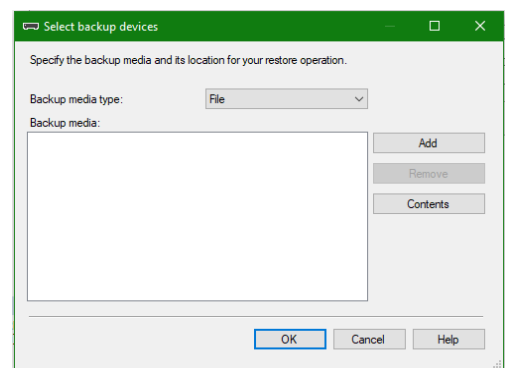
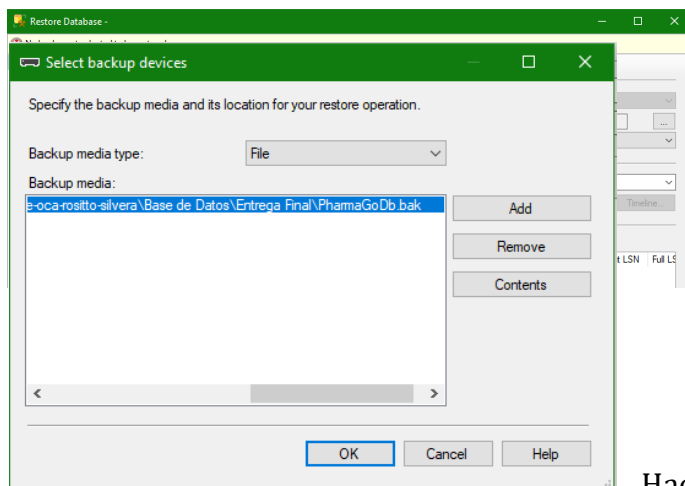
En el menú a la izquierda abrimos nuestra instancia de SQLEXPRESS, abrimos Database y hacemos click izquierdo, Restore Database



Seleccionamos Device y hacemos click en ...
y luego en Add

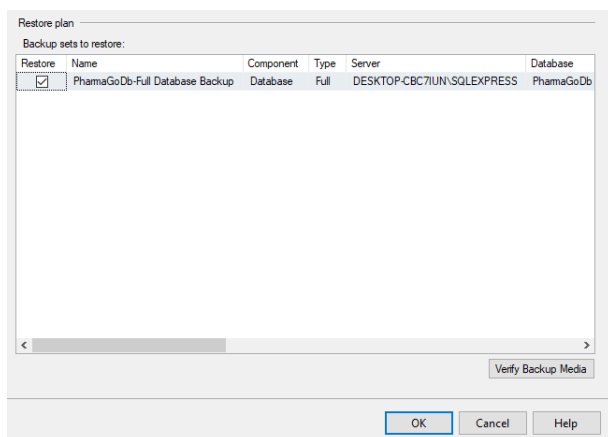
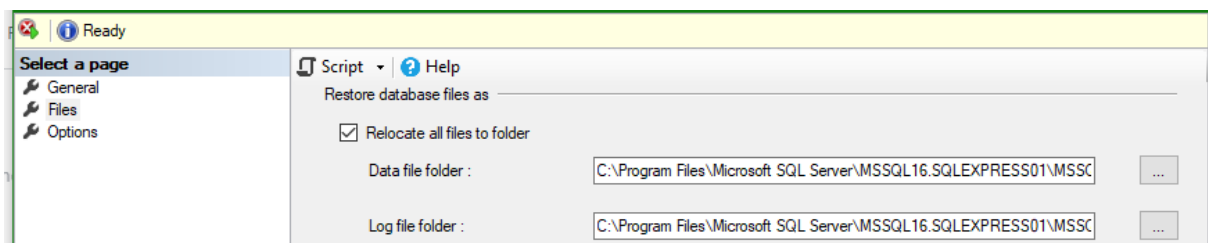
Buscamos nuestro archivo .bak ubicado en la carpeta del repositorio

Bases de Datos/ Entrega Final

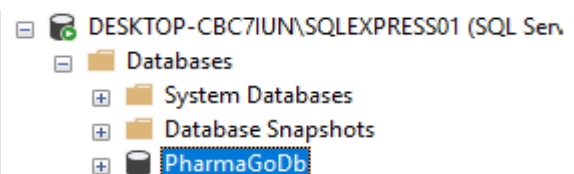


Hacemos click en OK

Y accedemos a la seccion Files a la izquierda y tildamos la opción Relocate all files to folder



Finalmente hacemos click en ok y nuestra base de datos estará lista para funcionar en conjunto con nuestro backend



corriendo en el administrador de tareas

Asegurarse que nuestro servicio este