Universidad Simón Bolívar
Dpto. de Computación y Tecnología de la Información
CI-3715 Laboratorio de Ingeniería de Software I
Enero-Marzo 2017

# SIGPAE- Históricos

**Equipo PowerSoft:**
Bárbara Hernández,10-11246
Domingo Arteaga,11-10058
Edgar Silva,11-10968
Edwar Yépez, 12-10855
Melanie Gomes ,13-10544
Verónica Mazutiel, 13-10853
**Profesores**: Betzaida Romero y Alfonso Reinoza

Sartenejas, 31 de Marzo de 2017

**ÍNDICE GENERAL**

## INTRODUCCIÓN

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, hemos adoptado este método para la realización del proyecto de la materia de ingeniería de software (SIGPAE) y en definitiva este fue pieza fundamental para la elaboración de la entrega final del mismo proyecto.

En la primera sección “Descripción del sistema” se explicará de manera detallada las bases y fundamentos del sistema desarrollado (SIGPAE-Histórico) con el fin de informar a todo aquel que esté interesado en evaluar el proyecto suministrado.

En la sección dos “Historias de usuario”, se describe con detalle la implementación de cada una de éstas. Se explica las estrategias, librerías y métodos usados, para logar el objetivo y finalizar la historia. Además, para cada historia se analiza el dominio de datos.

Luego, en la sección tres, “Experiencia con TDD”, se relata brevemente las pruebas unitarias realizadas para verificar que las historias de usuarios han sido desarrolladas de manera correcta, y la experiencia y beneficios que nos trajo para entregar un producto final de calidad que nos proporcionó este método..

En la sección cuatro, “Manejo del repositorio GitHub” se define cómo fue el uso del repositorio para cada integrante del equipo y de qué manera nos pudo ayudar y afectar para la integración de los códigos realizados por cada miembro, además, las ramas donde se ha venido trabajando durante el trimestre, y en donde se encuentra el resultado final del SIGPAE-Histórico elaborado por el equipo.

Asimismo, en la sección de “Conclusiones y Recomendaciones”, se expresa la experiencia en la elaboración del sistema, cómo fue la misma, y que recomendaciones se le añaden a los estudiantes -y profesores- que continuarán con el proyecto.

Finalmente, se especifican algunas fuentes consultadas para la realización del informe y del sprint, en la sección “Referencias”.

En el transcurso de los sprints entregados en el trimestre actual, fueron desarrolladas 17 de las historias de usuarios determinadas por el product owner, siendo importante acotar que a pesar de no haber desarrollado las 20 en su totalidad, la funcionalidad de las implementadas es de un 100%, ya que además de haber aprobado satisfactoriamente una cantidad significativa de pruebas (mencionadas en la sección “Experiencia con TDD”), estas fueron desarrolladas de manera eficiente, ya que se realizó un estudio exhaustivo de cada una de las herramientas que servían para cada una de ellas, determinando cuál era la más apta para que las historia de usuario fuese funcional, útil y segura, asegurándonos de que el sistema pueda perdurar en un futuro, y que sea de gran beneficio para nuestra casa de estudios la USB.

Para el desarrollo eficiente y correcto de cada una de ellas fue sin duda indispensable la organización y el buen manejo de las herramientas suministradas, como GitHub y Django, por otra parte, luego de todas las reuniones sostenidas con nuestra Scrum Master, hemos determinado que la

manera en la que ejecutamos los sprints anteriores fue idónea, sin embargo, siempre es posible superarse y mejorar, por lo cual para esta entrega continuamos con las mismas técnicas, pero buscando aumentar el rendimiento, para lograr constantemente la aprobación del Scrum master y por ende la aceptación del Product Owner.

## 1. Descripción del sistema.

**Perspectiva del sistema:**

El SIGPAE-HISTÓRICO implementado por el equipo será un sistema web que se centra básicamente en automatizar y digitalizar el proceso de elaboración de los programas análiticos de la Universidad Simón Bolívar, evitando de esta manera posibles errores de transcripción, y facilitándole a todos el personal que hace vida en el campus (estudiantes, profesores, coordinadores, etc) el acceso a estos.

Para que el sistema sea válido y que se pueda aprobar su uso, fue necesario seguir a cabalidad y de manera estricta las exigencias del product owner, ya que este es el que determinó en conjunto con DACE y los reglamentos de la universidad, que campos eran necesarios, obligatorios y opcionales a la hora de rellenar un programa análitico para nuestra casa de estudios.

El sistema cuenta con distintas opciones, empezando por cargar el pdf (imagen o texto), y elegir si quieren que sea extraído a texto de manera sencilla, o que determine el código y el departamento de manera automática, sin embargo, al escoger la segunda opción se le da la oportunidad al usuario mediante un mensaje de alerta que verifique si el código extraído en conjunto con el departamento asociado son los correctos, y en caso de que el código encontrado no se encuentre almacenado en la base de datos, se le da la opción de añadirlo y agregar el departamento al cual pertenece la asignatura.

Por otra parte, una vez extraído el texto, el usuario podrá rellenar cada campo correspondiente (obligatorios) para la elaboración del programa, sin embargo, de ser necesario, si no completa los recaudos, las modificaciones queda almacenada como “transcripciones en curso” ofreciéndoles así a los usuarios poder continuar sin ningún tipo de inconvenientes con la transcripción.

Además, el usuario está en la capacidad de generar nuevos campos que considere relevantes para la elaboración del programa a transcribir, de manera que el sistema no presenta rigidez a la hora de cambios en las asignaturas.

Es importante añadir, que el transcriptor puede proponer o solicitar que está transcripción pase a ser un programa analítico, luego del llenado de sus datos para conocer quien es el autor del mismo, haciendo esto totalmente eficiente a la hora de revisar responsables de los nuevos programas analítcos elaborados en nuestra universidad.

Por último pero no menos importante, se tiene acceso a los programas analiticos ya transcritos por el sistema SIGPAE, siendo posible realizar consultas a los mismos.

**Resumen de las capacidades del sistema:**

| Necesidades del product owner. | Prioridad. | Rango de calidad Realizada. |
|---|---|---|
| Convertir pdf a texto | ALTA | 100% |
| Recortar y pegar textstrings en áreas editables. | ALTA | 100% |
| Editar textstrings en áreas editables de la pantalla. | ALTA | 100% |

| | | |
|---|---|---|
| Mostrar todos los campos estándares asociados a programas analíticos | ALTA | 100% |
| Consultar programas analíticos almacenados en SIGPAE | ALTA | 100% |
| Registrar período en que entra en vigencia un programa | ALTA | 100% |
| Extraer texto de PDF de imágenes. | MEDIA | 100% |
| Salvar estado de transcripción. | MEDIA | 100% |
| Seleccionar instancia responsable del programa | MEDIA | 100% |
| Seleccionar departamento del programa. | MEDIA | 100% |
| Extraer código de asignatura y departamento | MEDIA | 100% |
| Manejar campos adicionales | BAJA | 100% |
| Separar fuentes de información recomendadas. | BAJA | 0% |
| Optar por alternativa a Fecha de entrada en vigencia. | BAJA | 100% |
| Solicitar convertir una transcripción en programa analítico propuesto | BAJA | 100% |
| Optar por alternativa a campo Objetivos | BAJA | 100% |

**Licencias e instalación.**

Para la instalación del SIGPAE-Histórico no se necesita ningún tipo especial de licencias, debido a que tanto Django (framework utilizado) como Python (lenguaje de programación utilizado) son de libre distribución.

Para instalar el programa se debe tener acceso a nuestro repositorio en Git-Hub y descargar la carpeta master, donde está contenido el 100% del desarrollo a lo largo del trimestre, además es necesario para su correcta ejecución tener instalado el framework Django y las herramientas de python : xpdf, pyocr, pdfminer,PIL librería psyscop2 de python y wand. Por último, es indispensable para las siguientes etapas del programa contar con postgresql, ya que es el lenguaje de base de datos utilizado.

A continuación recomendaciones para la instalación del sistema:

➔ Para instalar PDFMiner:
- Se puede descargar en https://pypi.python.org/pypi/pdfminer3k.
- Descomprimir el archivo descargado.
- Moverse a la carpeta descomprimida en el terminal y ejecutar python setup.py El proyecto podrá ahora usar la herramienta PDFMiner.

➔ XPDF:Se puede descargar escribiendo en el terminal:
- sudo apt-get update
- sudo apt-get install xpdf

➔ Descargar pyocr, wand y PIL para python3.

➔ Librería psycop2 para python3.

➔ Además,podría ser necesario instalar dj-database-url :
- Escriba en el terminal: sudo pip3 install dj-database-url

Por último, para la conexión con SIGPAE es necesario crear una base de datos en postgreSQL. Para ello habrá postgres en su consola y ejecute : create user sigpae with password '123123'; createdb sigaedb with owner sigpae. Finalmente ejecute los scripts SIGPAEsqchema.sql , SIGPAEdatos.sql , SIGPAEdatos2.sql que se encuentran en el repositorio.

**Estándares aplicados para el desarrollo del sistema.**

Para los requisitos de la asignatura estaba previsto que el sistema estuviese implementado en el lenguaje python con uso del framework Django, ya que este se adapta de manera adecuada a las necesidades del product owner, además, el mismo facilita el desarrollo web haciéndolo más legible y entendible, de manera que se pueda trabajar eficientemente.

Los estándares aplicados son:

- Utilización del lenguaje python.
- Utilización del frameWork Django
- Postgresql como manejador de bases de datos.
- Desarrollo por el método TDD.
- Utilización del modelo MTV.

**Requerimientos del sistema.**

Para que el sistema esté activo, se necesita cumplir ciertos requisitos descritos a continuación:

- Contar con un sistema operativo: Windows XP, Windows Vista, Windows 7, Linux (Ubuntu, Mint, Debian, o cualquiera de sus miembros).
- Conexión a internet
- Servidor web.
- Memoria RAM: el computador debe contar con memoria suficiente para acceder a la página.
- Debe asegurarse el respaldo de toda la información seguidamente en otros dispositivos de almacenamiento.
- Tener instaladas las herramientas de python : xpdf, pyocr y wand.

## 2. Historias de Usuario

### 1.1. Convertir PDF a texto

→ Descripción de la historia:

Como transcriptor necesito que el sistema me convierta un programa analítico que se encuentra a partir de una página web en un archivo PDF, en un texto (textString) para evitar que la transcripción necesaria para incluir programas existentes en un repositorio tenga que hacerse manualmente.

Pruebas de aceptación:

- Si el transcriptor selecciona un archivo y éste no existe, el sistema debe arrojar un mensaje de error y permitir seleccionar otro archivo o cambiar de página web.
- Seleccionado el archivo, no es un archivo PDF. El sistema debe arrojar un mensaje de error y permite seleccionar otro archivo o cambiar de página web.
- Seleccionado un archivo PDF, el sistema debe colocar en pantalla el textstring que resulte de su conversión. Las palabras en el textstring deben quedar separadas al menos por un carácter separador como espacio en blanco, nueva línea o signo de puntuación y debe respetar los caracteres propios del Castellano (acentos, diéresis, eñe…).

Para la implementación de esta historia de usuario utilizamos la herramienta xPDF de python, con la cual pudimos hacer la conversión de pdf a texto. Asimismo, se desarrolló una interfaz destinada a hacer la carga de un archivo.

El dominio de datos son todos aquellos archivos con la extensión .pdf y se verifica si el pdf existe y si realmente es un pdf. Se tiene un archivo test.py donde se realizan todas las pruebas unitarias para cada historia de usuario.

### 1.2. Recortar y pegar textstrings en áreas editables

→ Descripción de la historia:

Como transcriptor necesito recortar la subcadena que escoja de un textstring en un área editable de la pantalla y pegarla en cualquier otra área editable de la pantalla que contenga textstrings. Tales áreas están asociadas a la conversión de un archivo PDF o a campos como “nombre de la asignatura”, “código de la asignatura”, “créditos de la asignatura” u “objetivos”. Necesito esto para estructurar adecuadamente el programa analítico, evitando tener que tipear manualmente el contenido de cada campo.

Criterios de aceptación:

- Al recortar la subcadena, ésta se elimina del textstring, quedando el cursor marcando la posición donde se podría insertar la subcadena en el textstring nuevo para recuperar el textstring previo al recorte.
- Un textstring recortado puede insertarse en cualquiera de las áreas asociadas a campos de textstring, incluyendo el área de donde se recortó.

Para la implementación de esta historia de usuario su uso un elemento de tipo formulario del tipo output en donde se coloca la conversión del pdf. El texto que allí se muestra es posible de recortar y pegar en otras áreas implementadas en la interfaz del programa. Todo esto conectándose con los modelos del proyecto de django.

El dominio son strings que son manejados automáticamente.

### 1.3. Editar textstrings en áreas editables de la pantalla

→ Descripción de la historia:

Como transcriptor necesito desplegar, ver y editar un textstring en un área de la pantalla para poder corregir manualmente cualquier posible error en la conversión o su procesamiento posterior.

Inicialmente la edición se limita a mover el cursor por el textstring, así como agregar y eliminar caracteres a partir del cursor, recordando que se debe poder manejar los caracteres típicos de un texto en castellano o inglés (esto es incluyendo acentos, eñes y diéresis).

Criterios de aceptación:

- El área de despliegue puede contener una cadena vacía de caracteres. En este caso el sistema permite ingresar caracteres.
- El cursor puede moverse antes del primer carácter del textstring, después del último o entre cualquieras dos caracteres consecutivos del textstring.
- Se borra el carácter que queda a la izquierda del cursor. Un cambio de línea se considera un carácter.
- Al tipear un carácter se agrega el nuevo carácter a la derecha del carácter que quedaba inmediatamente a la izquierda del cursor y el cursor se posiciona después del carácter recién ingresado.

Esta historia de usuario está asociada con la historia de usuario previa, pues al contar con las áreas editables donde es posible pegar el textstring que se convirtió, en estas mismas áreas se puede editar el texto allí escrito.

El dominio son strings que son manejados automáticamente.

**1.4. Seleccionar departamento del programa**

→ Descripción de la historia:

Como transcriptor necesito seleccionar de un menú el nombre del departamento académico asociado al programa para registrar la responsabilidad departamental del programa.

NOTA: Esta historia fue sustituida por la 11.

**1.5. Registrar período en que entra en vigencia un programa**

→ Descripción de la historia:

Como transcriptor necesito registrar el período a partir del cual es vigente el programa. El código de una asignatura y el período en que entró en vigencia identifican unívocamente cualquier programa analítico en la Universidad Simón Bolívar por lo que registrar tal período es crucial en la construcción de un repositorio de programas analíticos.

Criterios de aceptación:

- Para identificar el período de entrada en vigencia de un programa se especifica el trimestre (sep-dic, ene-mar, abr-jul, intensivo) y el año (cuatro dígitos).
- No se permite introducir años previos a 1969, año que identifica a la primera cohorte que ingresó a la USB.

Esta historia de usuario debe contar con una base de datos para poder tener registrados los diferentes períodos que se pueden cursar en la USB.

El dominio de datos,cuenta con Enero-Marzo, Abril-Julio,Septiembre-Diciembre Intensivo. Para mostrarlos y escoger uno en la interfaz se colocó un menú desplegable.

**1.6. Extraer texto de PDF de imágenes.**

→ Descripción de la historia:

Como transcriptor necesito que el sistema extraiga texto de archivos en PDF resultado de haber convertido un programa analítico en una imagen para evitar que la transcripción necesaria para

incluir programas existentes en un repositorio tenga que hacerse manualmente.

Pruebas de aceptación:

- Si el transcriptor selecciona un archivo y éste no existe, el sistema debe arrojar un mensaje de error y permitir seleccionar otro archivo o cambiar de página web.
- Si se selecciona un archivo que no está en formato PDF, el sistema debe mostrar un mensaje de error y permitir seleccionar otro archivo o cambiar de página web.
- Seleccionado un archivo PDF, el sistema debe colocar en pantalla el textstring que resulte de su conversión. Las palabras deben quedar separadas al menos por un carácter separador como espacio en blanco, nueva línea o signo de puntuación y debe respetar los caracteres propios del Castellano (acentos, diéresis, eñe…).

Para el desarrollo de esta historia de usuario utilizamos las herramientas pyocr y wand de pythom, con la cual pudimos hacer la conversión de pdf de imágenes a texto. Asimismo, se desarrolló una interfaz destinada a hacer la carga de un archivo, donde el usuario debe seleccionar si desea convertir pdf o pdf imagen.

El dominio de datos son todos aquellos archivos con la extensión .pdf y se verifica si el pdf existe y si realmente es un pdf.

### 1.7. Mostrar todos los campos estándares asociados a los programas analíticos.

→ Descripción de la historia:

Como transcriptor necesito que el sistema muestre -de manera agradable-, todos los campos estándar asociados a programas analíticos para que el transcriptor pueda visualizar los campos que puede llenar a partir de la información que le presenta el archivo PDF que transcribe.

Pruebas de aceptación

- Los campos estándar son los indicados en artículo 3 del Reglamento de Administración de los Programas de Estudios de Pregrado a saber:
  - Código [Tipo de campo a determinar]
  - Denominación [Campo de texto]
  - Fecha de entrada en vigencia [Campo que permite un período en los términos establecidos por la Historia de Usuarios 5]
  - Horas de teoría, horas de práctica y horas de laboratorio. La suma de los tres tipos de horas está entre 0 y 40 horas semanales. Puede quedar nulo en cuyo caso se muestra un espacio en blanco, cuyo valor numérico asociado es 0 (cero).
  - Unidades créditos. Una asignatura puede tener de 0 a 16 créditos. Puede estar en blanco.
  - Requisitos [Tipo de campo a determinar]
  - Objetivos [Campo de texto, deslizable, ajustable]
  - Contenidos sinópticos [Campo de texto, deslizable, ajustable]
  - Estrategias metodológicas [Campo de texto, deslizable, ajustable]
  - Estrategias de evaluación [Campo de texto, deslizable, ajustable]
  - Fuentes de información recomendadas [Tipo de campo a determinar, deslizable, ajustable]

  y adicionalmente:

  - Departamento

- Los campos están claramente etiquetados.
- El transcriptor puede deslizarse (¿sólo verticalmente?] sobre el área que contiene los campos.
- Sería deseable que el área de los campos sea visualmente igual al proporcionado por SIGPAE.

Además de los campos de departamentos, título, código, período que entra en vigencia ya habilitados en las historias anteriores, se añadieron nuevos campos asociados a los programas analíticos, estos son:

- Horas de teorías y de práctica (Semanales).
- Unidades de créditos.
- Objetivos.
- Requisitos.
- Contenidos sinópticos.
- Estrategias metodológicas.
- Estrategias de evaluación.
- Fuentes de información recomendadas.

El dominio de esta historia de usuario se determina por cada campo, en el caso de las horas de teoría y práctica el dominio son enteros, y la suma de ambos debería ser menor o igual a 40 horas, por su parte, para las unidades de créditos también tienen un dominio de enteros, pero en este caso están limitados a ser números entre 0 y 16, finalmente el resto de los campos tienen un dominio de strings.

### 1.8 Consultar programas analíticos almacenados en SIGPAE.

→ Descripción de la historia:

Como usuario de SIGPAE-Histórico necesito consultar los programas analíticos de estudios contenidos en el repositorio de SIGPAE para leerlos, chequear si un programa analítico en PDF ya ha sido transcrito y, eventualmente, usarlo como como punto de partida para la transcripción de un programa similar.

Pruebas de aceptación

- El usuario debe introducir el código de la asignatura y el período (trimestre, año) para el cual busca el programa vigente para la fecha.
- El usuario puede omitir el período en cuyo caso el sistema le devolverá una lista de los programas asociados a esa asignatura ordenados por orden decreciente de período.
- El sistema debe devolver el programa analítico cuyo período de entrada en vigencia sea igual al período en que se aprobó, o si no existe cuyo período de entrada sea la más reciente y previa de las que estén contenidas en el repositorio.
- El sistema debe generar un mensaje de error si no existe un programa para el código suministrado y permitir solicitar otro programa.
- El sistema debe generar un mensaje de error si no existe un programa vigente asociado al código para el período suministrado.
- El sistema debe recuperar del repositorio todos los campos almacenados asociados al programa analítico y mostrarlos en las áreas de pantalla debidamente etiquetadas asociadas a esos campos. Todos los campos deben estar incluidos en un área deslizable (scrollable).
- La consulta no modifica el programa analítico almacenado en el repositorio de SIGPAE.

Para la implementación de esta historia de usuario, el equipo creo tablas en nuestra base de datos con la información contenida en SIGPAE para así simular la conexión entre ambos sistemas, para así hacer posible las consultas de los programas analíticos ya almacenados allí.

El dominio viene dado por la base de datos ya implementada por SIGPAE.

**1.9. Salvar el estado de transcripción**

→ Descripción de la historia:

Como transcriptor necesito salvar el estado de una transcripción para evitar tener que reiniciar la transcripción cada vez que reinice la sesión y para retomar la transcripción con el mínimo retrabajo posible.

Pruebas de aceptación

- Un transcriptor puede tener múltiples transcripciones salvadas. Al decidir retomar una transcripción debe indicar cuál de las salvadas quiere retomar.
- Los datos incluidos en todos los campos se salvan; también se salva el archivo PDF y el estado más reciente del texto o HTML que ha sido extraido y trabajado en la sesión. Al retomar la transcripción se recuperan todas los datos de las áreas de trabajo (campos, PDF, texto o HTML extraido).
- Debe advertirse al transcriptor si intenta salvar una transcripción cuyo código y período de entrada en vigencia corresponde exactamente a un programa que ya se encuentra en el repositorio de SIGPAE.
- Se puede salvar una transcripción al que le falten datos en cualquiera de sus campos. Para salvar una transcripción el mínimo estado que se debe tener es el PDF sobre el cuál se trabaja.
- Las transcripciones salvadas no deben confundirse con programas analíticos, una transcripción salvada representa, en todo caso, el borrador de un programa analítico en estado incompleto de construcción.

Para esta historia, básicamente lo que se realizó fue colocar un botón de transcripciones en curso, donde se pueden ver todas las transcripciones elaboradas previamente, para continuar su modificación o edición en el momento deseado.

Esta historia tendría como dominio el conjunto de transcripciones no finalizadas con la información que se guardó en sus campos estándares junto a la transformación del pdf correspondiente al programa.

**1.10. Seleccionar instancia responsable del programa.**

→ Descripción de la historia:

Como transcriptor necesito seleccionar de un menú el nombre del ente responsable de dictar el programa analítico para que quede registrada tal responsabilidad del programa.

Criterios de aceptación:

- El mecanismo de selección debe permitir seleccionar cualquiera de los departamentos académicos adscritos a las cuatro divisiones de la USB (Ciencias Físicas y Matemáticas, Ciencias Sociales y Humanidades, Ciencias Biológicas, Ciencias y Tecnologías Administrativas e Industriales), así como cualquiera de las Coordinaciones académicas adscritas a los Decanatos de Estudios (Estudios Generales, Estudios Tecnológicos, Estudios Profesionales, Postgrado). Generalmente los departamentos son responsables de un

programa analítico pero existen un pequeño número de asignatura por las que son responsables las coordinaciones (p. ej. Proyecto de Grado, Pasantía).

- El mecanismo de selección debe ser fácil y agradable de usar. Por ejemplo, no puede ser un menú que presente más de doce opciones.
- El nombre de cada instancia debe estar completo (p.ej. Computación y Tecnología de la Información)
- [A verificar, puede no ser cierto en caso de las coordinaciones] Un programa es responsabilidad de exactamente una instancia.

Esta historia de usuario debe contar con una base de datos para poder tener registrados los departamentos y coordinaciones al que puede pertenecer el programa de estudios, para esto, a nivel de interfaz, existen dos casos, el primero es que el usuario seleccione como encargado un departamento donde se desplegarían los distintos departamentos existentes en la USB, por su parte, si el usuario selecciona como encargado a una coordinación, se muestra la lista de las coordinaciones pertenecientes a la academia.

El dominio de datos, son todos aquellos departamentos y coordinaciones adscritos a todas las divisiones de la universidad, para mostrarlos y escoger uno en la interfaz se colocó un menú desplegable. Para cargar todas estas instancias utilizamos un archivo .json con dicha información.

### 1.11 Extraer código de asignatura y departamento.

→ Descripción de la historia:

Como transcriptor me interesa activar la opción que el sistema proponga un código de asignatura y un departamento (o instancia) y me permita cualquier corrección al respecto para reducir la tasa de errores que podrían presentarse en una transcripción netamente manual.

Criterios de aceptación:

- El sistema incluye un menú denominado “Extraer”, una de cuyas opciones es “código y dpto”.
- El código de la asignatura puede seguir la palabra clave “Código” o la frase “código de asignatura” o simplemente aparecer como parte de la denominación de la asignatura como por ejemplo “ASIGNATURA: FS2233 – Ondas y Óptica”
- Un código de asignatura puede tener cualquiera de las siguientes formas: (a) Dos letras seguidas por cuatro dígitos, (b) tres letras seguidas por tres dígitos (Estudios Generales, algunas asignaturas viejas como MAT100). Puede haber un guión o un blanco entre las letras y los dígitos que se ignora (MA-1111, CI 4121, CSA 211).
- En el apéndice A se anexa una tabla (incompleta) de las siglas asociadas a las diferentes dependencias.
    - En caso que el sistema encuentre en el programa analítico un código de asignatura cuyas siglas se encuentren en la tabla asociada a alguna dependencia (típicamente departamento), el sistema reportará cuál y propondrá, en el campo correspondiente, el departamento actual al que está adscrito, dejando propuesto el código en el campo de código.
    - En caso que el sistema encuentre las siglas en la tabla correspondiente pero no están asociadas a una dependencia (EP, ET, PG, PD, TG, TEG, USB…), el sistema reportará que se trata de siglas válidas asociadas a lo que indique la tabla (por ejemplo “proyecto de grado”), propondrá el código en el campo correspondiente y dejará en blanco el

campo de la dependencia a que está escrito.

- En caso que el sistema no encuentre las siglas en la tabla correspondiente, pero el código cumpla con la estructura de código (dos letras seguidas por cuatro dígitos, o tres letras seguidas por tres dígitos), el sistema indicará que las siglas son desconocidas y preguntará al transcriptor si quiere proponerlas. En caso que el transcriptor responda afirmativamente, el sistema le presentará una lista de opciones que incluyen los departamentos del apéndice A, y una opción que indique "ninguna de las anteriores". Si el transcriptor escoge "ninguna de las anteriores", el sistema dejará que el transcriptor ingrese siglas que no estén en la tabla, el sistema marcará y registrará las nuevas siglas como "propuestas", por lo que eventualmente estarán sujetas a la aprobación de un ente autorizador. Si el transcriptor escoge "ninguna de las anteriores" pero introduce siglas que ya están en la tabla, el sistema las aceptará, advirtiendo al transcriptor la asociación que tienen (pero no se marcan como "propuestas").

- En caso que el sistema no encuentre en el programa analítico un código de asignatura, debe reportarlo y darle la posibilidad al transcriptor de introducir manualmente el código (2 letras, un símbolo que puede ser letra o dígito, dos dígitos. En caso que introduzca unas siglas que no estén en la tabla, el sistema advertirá esto al transcriptor y solicitará al transcriptor que confirme las siglas. De confirmarlas, el sistema marcará y registrará las nuevas siglas como "propuestas" por lo que eventualmente estarán sujetas a la aprobación de un ente autorizador.
- El sistema reportará el primer código válido que encuentre.
- Al salvar la transcripción, el sistema salva las marcas de siglas "propuestas" que pudieran haberse generado, las cuáles se harán visibles al restaurar la transcripción.

Para el desarrollo de esta historia, se analiza el texto transformado del pdf y se compara con una expresión regular que delimite la estructura de los códigos de la universidad, el primer código que coincida con esta expresión regular se guarda en el campo código, sin embargo, sigue siendo posible editarlo si el código encontrado no resulta el correcto, luego de conocer las dos o tres letras con las que inicia dicho código, se realiza una búsqueda en una tabla que contiene todas las iniciales de materias con su respectivo departamento, para determinar el departamento al cual pertenece esta asignatura, sugiriéndoselo al usuario para ser almacenado en el campo departamento, si no coincide con ningún código adscrito a algún departamento el usuario tiene la posibilidad de proponerlo.

El dominio de esta historia de usuario viene dado por strings, ya que la funcionalidad consiste en una búsqueda en el texto transformado.

**1.12. Manejar campos adicionales.**

→ Descripción de la historia:

Como transcriptor quiero capturar información que aparece en un programa analítico cuya estructura no se corresponde con lo indicado en el reglamento para no perder información que puede resultar útil. Para ello necesito manejar de manera más flexible campos adicionales.

Criterios de aceptación

- Los campos se dividen en tres categorías: obligatorios, normados y adicionales. Como mínimo, un programa contiene los campos obligatorios a saber: Código, Departamento, Denominación, Fecha de entrada en vigencia, Horas de teoría, horas de práctica y horas de laboratorio, Contenidos sinópticos, Fuentes de información recomendadas

El reglamento de administración de estudios (pregrado) explicita campos adicionales, que se denominarán normados, que muchos programas analíticos -particularmente los aprobados previos al

reglamento correspondiente vigente- no incluyen. Los campos normados incluyen: Requisitos, Objetivos, Estrategias metodológicas, Estrategias de evaluación.

Cualquier otro campo (por ejemplo Justificación) se considera un campo adicional.

- En caso que el transcriptor encuentre un campo adicional que considere pertinente, puede activar una opción de "Agregar campo". En este caso el sistema le mostrará los campos adicionales registrados en el sistema, más la opción "Ninguna de los anteriores". Si el campo que busca corresponde a alguno de ellos puede escogerlo y el sistema lo agregará a los campos posibles para el programa que transcribe. Si el campo que busca no corresponde a ninguno de los registrados, el transcriptor escoge la última opción (Ninguno de los anteriores) y el sistema procede a agregar tal campo a la lista de campos adicionales bajo el nombre dado por el usuario (el sistema no permite que se dupliquen los nombres de los campos).
- Un campo adicional es un campo de texto.
- Al salvar una transcripción se salvan los campos adicionales que pudieran haberse introducido. Estos campos adicionales y su contenido se restauran al restaurarse la transcripción.

Para la implementación de esta historia de usuario se agregó una tabla en la base de datos de campos adicionales, y una de contenido extra, una vez desarrollado eso, en la vista del relleno de los campos del programa se agregar un botón de "agregar un nuevo campo", si este es presionado por el usuario se muestra un campo que podrá ser editado por el, siendo posible agregar tantos campos adicionales como se requiera, además, se verificará que los nombres de los campos que el usuario está agregando no estén repetidos. El dominio de datos son campos de texto.

**1.13. Optar por alternativa a Fecha de entrada en vigencia.**

→ Descripción de la historia:

Como transcriptor puedo escoger entre manejar la fecha de entrada en vigencia como un período y un año, o como una fecha para no tener que tomar la decisión de a qué período académico corresponde una fecha indicada en un programa analítico viejo.

Criterios de aceptación

- El transcriptor puede indicar solo el año de entrada en vigencia del programa, el mes y el año, o el día, mes y año, según opciones de tres menúes (día, mes, año).
- (Tentativo) Por defecto el sistema asociará fechas a períodos de la siguiente forma:
  - Fecha entre 1º enero y 31 enero de un año corresponde a entrada en vigencia en ene-mar del mismo año;
  - Fecha entre 1º febrero y 15 abril de un año corresponde a vigencia a partir de abr-julio de ese mismo año;
  - Fecha entre 16 abril y 30 septiembre de un año corresponde a vigencia a partir de sep-dic del mismo año;
  - Fecha entre 1º de octubre y 31 diciembre de un año corresponde a vigencia a partir de ene-mar del año siguiente.
  - El sistema marcará la fecha de entrada en vigencia como tentativa. La asociación definitiva la deberá fijar el Jefe de Dpto o el Coordinador de Carrera y ser ratificado por DACE.
- El transcriptor puede corregir la fecha de entrada en vigencia de un programa, en cuyo caso se recalcula el período de entrada en vigencia del mismo. También puede cambiar el período de entrada en vigencia, pero en tal caso no se cambia la fecha de entrada en vigencia, sino que el sistema marca tanto en la fecha como el período que no se corresponden.
- Al salvarse/restaurarse la transcripción, se salvan/restauran el período, la fecha de entrada en vigencia y la marca de inconsistencia en caso que ésta exista.

Para la implementación de esta historia de usuario, se realizó la agregación de un nuevo

atributo a la tabla de transcripciones denominado depfile que se refiere a una fecha de transcripción. Mostramos los campos en el display pdf (fecha de entrada en vigencia y periodo), y cuando se cambia la fecha de transcripción automáticamente se actualiza el periodo y el año de vigencia, en caso contrario, si se cambia el periodo no se cambia la fecha evaluando la consistencia de los datos. En caso de ser inconsistentes se muestra una alerta al usuario de que hay inconsistencia en este par de datos.

El dominio de esta historia son string de fechas, periodo y año.

**1.14. Solicitar convertir una transcripción en programa analítico propuesto.**

→ Descripción de la historia:

Como transcriptor necesito solicitar que una transcripción se registre en el SIGPAE como un programa analítico sujeto a aprobación por la instancia responsable para que los programas analíticos viejos que se transcriben puedan enriquecer el repositorio SIGPAE de programas analíticos.

Criterios de aceptación:

- Para ser aceptado como programa analítico sujeto a aprobación (PASA), todos los campos obligatorios deben contener valores no nulos.
- Para ser aceptado como PASA el código de la asignatura y el período en que entra en vigencia no puede ser la clave de ningún otro programa analítico que se encuentre en el repositorio.
- Para ser aceptado como PASA, la transcripción debe cumplir con las restricciones de sus campos.
- Al aceptar una transcripción como PASA, el sistema requiere que el transcriptor indique su nombre, correo electrónico y teléfono, los cuales registra.
- Un PASA deja visible para su aprobación explícita tanto siglas propuestas, campos adicionales, como asociaciones propuestas entre fecha de entrada en vigencia y período de entrada en vigencia
- Un PASA se guarda en el repositorio de SIGPAE –recuerde modificar esa base de datos para que pueda tomar en cuenta campos adicionales y los distintos tipos de marcas y la condición de PASA.
- (Temporalmente) Una vez aceptada una transcripción como PASA, la transcripción queda bloqueada, es decir no puede seguirse editando. (Eventualmente una futura historia explicará cómo se puede desbloquear)

Para la implementación de esta historia de usuario, lo que se realizó fue la agregación de dos booleanos a la tabla de transcripciones denominados pasa y propuesto, donde pasa sería True si todos los campos obligatorios están rellenados de forma correcta y formateada, por su parte, el atributo propuesto pasaría a True si alguien propone la transcripción como un programa analítico. Además, en la vista de todas las transcripciones realizadas se incorporó un botón “proponer” por cada transcripción que tenga sus datos completos, al seleccionar este, aparece un formulario para ser llenado con los datos de la persona que lo está proponiendo (nombre, apellidos, teléfono y correo) esto con el fin de poder acceder a los datos de la persona que propuso la transcripción como programa analítico. Para este último se construyó otra tabla llamada “transcriptor” con los mismos datos que introdujeron en el formulario.

El dominio de esta historia de usuario son string para rellenar el formulario al proponer las transcripciones.

**1.15. Optar por alternativa a campo Objetivos.**

→ Descripción de la historia:

Como transcriptor puedo seleccionar reemplazar el campo Objetivos por dos campos consecutivos Objetivos Generales y Objetivos Específicos para tener una estructura más flexible que

se corresponde con dos modalidades de especificar los objetivos en los programas analíticos.
Criterios de aceptación:

- En cualquier momento el transcriptor puede seleccionar una opción que pasa de una modalidad a otra,
- Al pasar de la modalidad con un solo campo (Objetivos) a la de dos campos (Objetivos Generales, Objetivos Específicos), el texto que se haya transcrito en el campo Objetivos pasa a formar parte del campo Objetivos Generales.
- Al pasar de la modalidad de dos campos (Objetivos Generales, Objetivos Específicos) a la modalidad de un solo campo (Objetivos), el sistema advertirá que se perderá la separación entre los campos, quedando toda la información en el único campo (toda el texto que se encuentre en Objetivos Generales seguido después de una nueva línea por todo el texto en Objetivos Específicos). El sistema procederá a hacer la operación solo si el transcriptor confirma que no le importa perder esa separación.
- Al salvar/restaurar una transcripción, se salva/restaura la modalidad vigente.
- Al solicitar convertir una transcripción en PASA, el repositorio de SIGPAE debe permitir almacenar la modalidad vigente al momento de la solicitud.

La implementación de esta se basó en la agregación de dos campos más en el formulario correspondiente a las transcripciones denominados “objetivos generales” y “objetivos específicos”, apareciendo el primero por defecto siempre para rellenar en el formulario,además, aparecerá un botón donde el usuario seleccionara (en caso de que requiera) la alternativa de separar los objetivos, en este momento aparecería la opción de rellenar el campo de objetivos específicos. Además, si el usuario quisiera volver a incorporar estos dos campos en uno, tendrá esta opción, y al presionarlas el sistema arroja una alerta donde se le pregunta si está seguro, en caso afirmativo, los datos introducidos en el campos objetivos específicos se colocan luego de lo escrito en objetivos generales.

El dominio para esta historia de usuario son strings.

**3. Experiencia con TDD**

A lo largo del tiempo se ha determinado la importancia que tiene el desarrollo dirigido por pruebas (TDD) a la hora de realizar un software, ya que esta es una práctica de programación que está fundamentada básicamente en escribir primeros las pruebas y en la refactorización del código, garantizando la estabilidad del producto. Primero se escriben las pruebas unitarias, seguido a esto, se genera el código fuente conociendo que pasará la prueba anteriormente elaborada para luego refactorizar este último código, todo esto trae como resultado un código más seguro, más robusto y más mantenible, siendo esto muy importante a la hora de elaborar un software de calidad.

En nuestro caso, fue indispensable aplicar la práctica de TDD para la implementación de SIGPAE-Histórico, ya que al realizar las distintas pruebas pudimos mejorar el código, y evitar errores o fallas encontradas en el código del mismo.

En un archivo llamado test.py se escribieron entre otras estas pruebas unitarias para determinar la eficacia de nuestro software:

- Se desarrolló algunas pruebas con la base de datos, las cuales generan otra base de datos, en donde se hacen las pruebas y luego son destruidas por unittest.
- El pdf escogido para convertir es realmente un documento con la extensión .pdf. Se hicieron pruebas de malicia con archivos .pdf , faltando únicamente la f.
- Se verificó que lo que se guardaran los departamentos en la base de datos coincidieran con cómo se crearon. Aquí no hay riesgo de errores por parte de los usuarios.
- Se verificó que lo que se guardaran los períodos en la base de datos coincidieran con cómo se crearon. Aquí no hay riesgo de errores por parte de los usuarios.
- Se verificó para cada campo, que el input ingresado por el usuario se guardará exitosamente en la base de datos y su correctitud.
- Se verificó que los acrónimos de la base de datos estuvieran correctos, evitando elementos basura.
- Se verificó que las unidades de créditos estuvieran en el dominio entre 0 y 16 créditos.
- Se verificó que las horas semanales (Suma de las horas de laboratorio y práctica) estuvieran en el dominio entre 0 y 40 horas
- .Se verificó la correcta agregación de los campos adicionales en la transcripción del programa analítico.

Sin embargo, si se quiere ver de manera detallada cada una de las pruebas implementadas, es recomendado acceder a el archivo test.py mencionado anteriormente.

Luego de la escritura de todas estas pruebas, pudimos determinar fallas en el código, y condiciones que no estábamos tomando en cuenta con respecto al dominio de datos, así se pudo corregir y verificar los detalles descritos, para luego refactorizar el código de manera que esté legible para cualquier persona del grupo o externa a él.

Por último, la experiencia obtenida por el desarrollo del software aplicando TDD a lo largo de los sprints ha sido sumamente satisfactoria, ya que hemos podido determinar cualquier tipo de falla que no se haya previsto en un principio para la escritura del código, además, luego de realizar las pruebas, se refactorizaba el código, siendo esto de alta importancia a la hora de verificar el código y de entenderlo.

## 4. Manejo del repositorio GitHub

Para manejar el sprint de manera conjunta, continuamos utilizando de manera eficiente el repositorio de GitHub que creamos para la primera entrega, en el que creamos distintas ramas para ir modificando los archivos sin dañar la rama master.

Fue necesaria la creación de varias ramas, que han ido cambiando a lo largo de las entregas. Se tienen las ramas Sprint1, Sprint2 y Sptin3 en donde se encuentra lo relacionado a dichas Sprints.

En el repositorio de GitHub se encuentran actualmente las ramas anteriormente mencionadas y Master, donde para cada una de ellas los miembros del equipo aportaron distintas funcionalidades, vistas, cambios en los modelos y correcciones de errores cuando era necesario.

El estado final de esta entrega se encuentra en la rama Master y para poder correr el proyecto se debe contar con Django e instalar las herramientas de python :xpdf, pyocr, pdfminer,PIL librería psyscop2 de python y wand.

Por otra parte, se ha certificado que sin el uso de esta herramienta (GitHub) el desempeño del equipo disminuiría, por lo cual, hacemos el mayor esfuerzo por usarla de la mejor manera, para sacarle todo el provecho posible.

## CONCLUSIÓN Y RECOMENDACIONES

Durante la realización de los sprints, pudimos percatarnos nuevamente de los beneficios de los distintos métodos de programación.

Las historias de usuario nos permitieron desarrollar de forma modular las funcionalidades de los sprints, así como implementar las mismas con una división de tareas bien específicas que luego pudieron ser mezcladas gracias a la herramienta Git.

Además hemos avanzado en el aprendizaje para el desarrollo usando el framework Django, así como a realizar interfaces gráficas e integrarlas con las funcionalidades del proyecto.

Los comentarios de nuestra Scrum Master, así como los emitidos por el Product owner, han sido claves fundamentales para continuar en la línea de eficacia, eficiencia y efectividad, ya que en su mayoría fueron alentadores y positivos con respecto a nuestro rendimiento.

Finalmente, concluimos que el trabajo en equipo con tareas bien específicas que a la vez pueden ser compartidas de manera organizada, consolida un buen desarrollo y finalización de un proyecto.

Por otra parte, las recomendaciones principales para este trabajo serían determinar para cada historia de usuario, su prioridad, su descripción y si es necesario la manera en la que fueron implementadas, también es relevante tener en cuenta todas la experiencia que se ha obtenido con el desarrollo del software aplicando la práctica de TDD, y como se ha manejado el repositorio de GitHub. Se puede decir que para poder manejar el software SIGPAE-Histórico no es necesario tener mucho conocimiento sobre el programa en sí, ya que está desarrollado de manera adecuada para que los usuarios se familiaricen con el sistema y puedan desarrollar lo esperado a través de él.

Para futuras extensiones del sistema se recomienda tener en cuenta que la base de datos de SIGPAE no está implementada de la manera correcta, ya que no hacen uso de claves foráneas, por lo cual cuando se requiere hacer referencia a otra tabla (que en teoria deberia ser por la clave foránea) se debe realizar por medio de joins, produciendo muy poca eficiencia y siendo muy engorroso a la hora de realizar consultas de gran tamaño, aumentando de manera considerable el costo de operaciones de entrada y salida y por ende, aumentando el tiempo de ejecución.

Por otra parte, todos los campos de las tablas son caracteres o booleanos haciendo que se inutilice la base de datos a la hora de realizar consultas más apropiadas con respecto a los campos. Además, la tabla correspondiente a la transición a programas tiene aproximadamente 39 atributos, siendo esto exagerado, ya que al momento de utilizar la tabla es necesario recordar cada uno de ellos, siendo prácticamente imposible, por lo cual siempre que sea necesario realizar consultas sobre está se debe tener acceso a la base de datos para evitar posibles errores con los nombres de los atributos o inclusive olvidar su existencia dentro de esa tabla. A parte de esto, todos estos atributos hacen referencia al estado de la solicitud más no a los programas como tal, ya que no contiene todos los campos obligatorios con los que debe contar cada programa analítico.

Además, está base de datos está implementada en un lenguaje que no es compatible con Django, que adicional a esto no conocemos y por ende no manejamos, por lo cual para poder cumplir con la historia referente a la conexión con está base de datos fue simulada la conexión con la misma creando una nueva con todas las tablas necesarias para resolver los problemas que se nos presentaron en la historia correspondiente.

Por último sobre la conexión a la base de datos de SIGPAE, se expresa la inquietud de

porque nuestro sistema debe acceder a está, ya que este sistema no paso la aprobación de DACE, por lo tanto, los programas dentro de está base de datos no son aprobados por los entes superiores de nuestra universidad, por lo que pueden considerarse no legales.

La recomendación fundamental para la posible expansión y continuación de este proyecto es conocer con detalle todas las bondades que ofrece el framework Django y el lenguaje de gestión de base de datos postgres, para así poder seguir construyendo de manera eficiente este sistema, además es fundamental informarse e investigar cómo funcionan las herramientas utilizadas por este equipo para cumplir con las historias de usuarios.

## REFERENCIAS

Documentación de python: https://docs.python.org/3/

Documentación de Django: https://docs.djangoproject.com/en/1.10/

Definición de Scrum: https://proyectosagiles.org/que-es-scrum/

Abad, S. “Lineamientos sobre cómo escribir informes técnicos”.(2007)