Trabajo Práctico Integrador.

Gestor de mails.

Integrantes: Suarez Joaquin
Briend Andres

Profesor/a: Fernandes Jose A.

Cátedra: Paradigmas y Lenguajes de Programación II **Cuatrimestre:** Segundo

Carrera: Ingeniería en Sistemas de la Información

Curso: 2do año **Año lectivo:** 2022

UNIVERSIDAD DE LA CUENCA DEL PLATA

Facultad de Ingeniería y Tecnología

Universidad de la Cuenca del Plata

Resumen

En el presente informe se buscará mostrar la resolución planteada por los alumnos, al escenario propuesto las consignas. En el informe se encuentran distintos tipos de información y partes del algoritmo desarrollado por los alumnos. Todo esto se encuentra dentro de los márgenes del contenido teórico y práctico que fue desarrollado en la cátedra de paradigmas de la programación II en el cuatrimestre cursado.

UNIVERSIDAD DE LA CUENCA DEL PLATA

Índice

UNIVERSIDAD DE LA CUENCA DEL PLATA

“Gestor de mails”

Presentación de Grupo

Para llevar a cabo el desarrollo del trabajo integrador, los integrantes de la cátedra optaron por trabajar de forma grupal. Se decidió por realizar para este trabajo en específico un grupo de tan solo 2(dos) estudiantes. Para realizar el desarrollo del gestor de mails solicitado.

Introducción

El problema planteado por las cátedra es el de que el alumnos pueda desarrollar un sistema de gestión de correo electrónico con el lenguaje de programación seleccionado previamente, también esté utilizando la técnica de la TDD. Es sistema de gestión de correo electrónico deberá implementar algunos requerimiento planteado en el escenario.

Detalle del algoritmo.

El algoritmo es capaz de simular un correo electrónico en el cual un usuario podrá enviar correos a uno o más usuarios. Estos correos o emails que se envían pueden tener un (Asunto,Contenido Y varios o un solo destinatario) permitiendo así que un usuario pueda enviar a varias o una personas un mismo correo. Un usuario también puede tener una lista de contactos en las cuales se almacenarán sus diferentes contactos los cuales son otros usuarios a los que él desea enviar dichos correos. Estos contactos guardan toda la información de un usuario (nombre, apellido y email o dirección de correo).

Un usuario tendrá una bandeja de salida de correos (en esta se almacenarán todos los correos en los cuales un usuario sea el emisor). También podrá tener una bandeja de entrada (en esta se almacenarán todos los correos en los cuales un usuario sea un receptor de un correo).

También un usuario podrá filtrar los correos que se encuentren en la bandeja a través de distintos filtros los cuales son ; Filtro por asunto (Filtra los correos fijándose en el asunto y lo hace a través de una cadena de texto que el usuario quiera buscar), Filtro por contenido (Filtra los correos fijándose en el contenido y lo hace a través de una cadena de texto que el usuario quiera buscar), Filtro por Emisor (Filtra los correos fijándose en el emisor y lo hace a través de una dirección de correo que el usuario quiera buscar). También existe la posibilidad de filtrar correos a través de dos campos ya sea, “emisor y asunto” o “emisor y contenido”. Los cuales también funcionan de una forma similar.

Los correos poseen un emisor, un asunto, el contenido del mismo, y varios o un solo receptor.

Desarrollo

Para el desarrollo del algoritmo decidimos dividir las tareas entre los integrantes del grupo, tomando en cuenta el área en que cada uno se encuentra mejor capacitado y más cómodo a realizar.

Para la realización de las tareas, cada integrante cuenta con su Rol respectivo y decidimos plantear las actividades a realizarse una a una, acordando una fecha aproximada en la que estas deberían estar finalizadas.

Sobre los principios "SOLID"

Los principios "SOLID" son un conjunto de principios que se pueden aplicar a lo que vendría siendo la programación orientada a objetos. Al aplicar estos principios de una forma correcta se puede decir que estaríamos escribiendo un código-Software de calidad en cualquier lenguaje de programación orientada a objetos.

¿Pero cuáles son estos principios "SOLID"?

- El principio de responsabilidad única o en inglés (Single Responsibility principle) ;

Este principio hace referencia a que un objeto debe realizar una única cosa logrando así tener clases más concisas que re respondan a una sola cosa, porque si no lo pensamos bien podríamos tener clases que terminan teniendo muchas responsabilidades lógicas innecesarias.

Este principio se podrá observar en nuestro algoritmo a la hora de ver las clases como “Filtro.java” o la clase “Gestordecontactos.java” las cuales tienen un objetivo único y claro dentro del algoritmo.

- Principio Open/Closed ;

Este principio nos hace referencia a que nuestro algoritmo tiene que estar abierto a extensiones pero cerrado a modificaciones. ¿Pero qué quiere decir con esto?
Esto hace referencia que una clase tiene que estar disponible a extender su comportamiento pero no se deberá de poder modificarlo. Todo esto sin tener la necesidad de modificar el código, de forma que podremos seguir añadiendo cosas sin afectar al código ya preexistente.

Este principio lo podemos observar en nuestro algoritmo en la clase “Usuario.java” se puede extender esta a nuevos métodos como por ejemplo: enviar archivos o videos. Sin tener la necesidad de modificar el código previamente desarrollado.

- Principio de sustitución de liskov o en inglés (Liskov substitution principle) ;

Este principio hace referencia a que en nuestro Software estamos usando una clase y si esta clase es extendida, Tendríamos que poder utilizar cualquier clase hija y el programa deba seguir siendo válido y funcionando.

Este principio se puede observar en el algoritmos con la clase “Correo.java” La cual es una clase hija heredera de una clase “CoreoBase.java”

- Principio de segregación de interfaces o en inglés (interface segregation principle) ;

Este principio hace referencia a que nuestro algoritmos no tendrá que existir clase que dependa de métodos que no usa. Entonces cuando creamos interfaces tenemos que estar seguros de que las clases utilizaran todos los métodos de las mismas. A veces es mejor tener varias interfaces más pequeñas. Ya que el uso de interfaces nos ayudan a desacoplar módulos.

Este principio se puede ver en en la implementación que hace la clase “GestorContacos.java” implementando a la interfaz “IBusquedas.java” siendo esta una interfaz pequeña pero super importante para esta clase en específico.

- Principio de inversión de dependencia o en inglés (Dependency inversion principle) ;

Cuando hablamos de este principio, hacemos referencia a uno de los principios más vistos o más importantes ya que cambiará nuestra forma de pensar a la hora de desarrollar código. Este principio nos dice que las clases de alto nivel no deberían de depender de las clases de bajo nivel. Ambas deberían depender de las abstracciones. Las abstracciones no deberían depender de los detalles si no que los detalles deberían de depender de las mismas.
O en otras palabras los atributos de una clase no tendrían que depender de otras clases si no de la abstracciones de las mismas, la cual se realiza utilizando la interfaz de una misma.

Esto se puede ver más claro en el algoritmo en nuestra clase “Usuario.java” la cual tiene como atributo una interfaz llamada “IAccionesBandeja.java” la cual es implementada por 2 clases, permitiéndonos así aplicar este principio en el algoritmo.

Objetivo grupal

Como grupo de trabajo nos propusimos como objetivo principal cumplir con todos los requerimientos solicitados en el escenario y agregar nuestra propia impronta al desarrollo de la solución, cumpliendo con todas o la gran mayoría de consignas. Realizan un desarrollo de la solución lo más limpio y claro posible para el entendimiento.

Participantes y Roles

Nombre y Apellido	Rol/Roles
Suarez, Joaquin	Codificación del Algoritmo/Desarrollo del informe // Documentación
Andres, Briend	Codificación del Algoritmo/Desarrollo del informe // Documentación

Actividades realizadas

Actividad
Diseño del Diagrama de Clases
Planteo del proyecto
Buscar información importante y necesario
Prototipo del Proyecto
Presentación del Proyecto
Ajustes al desarrollo
Pulir el algoritmo
Conclusiones
Crear informe
Crear power point

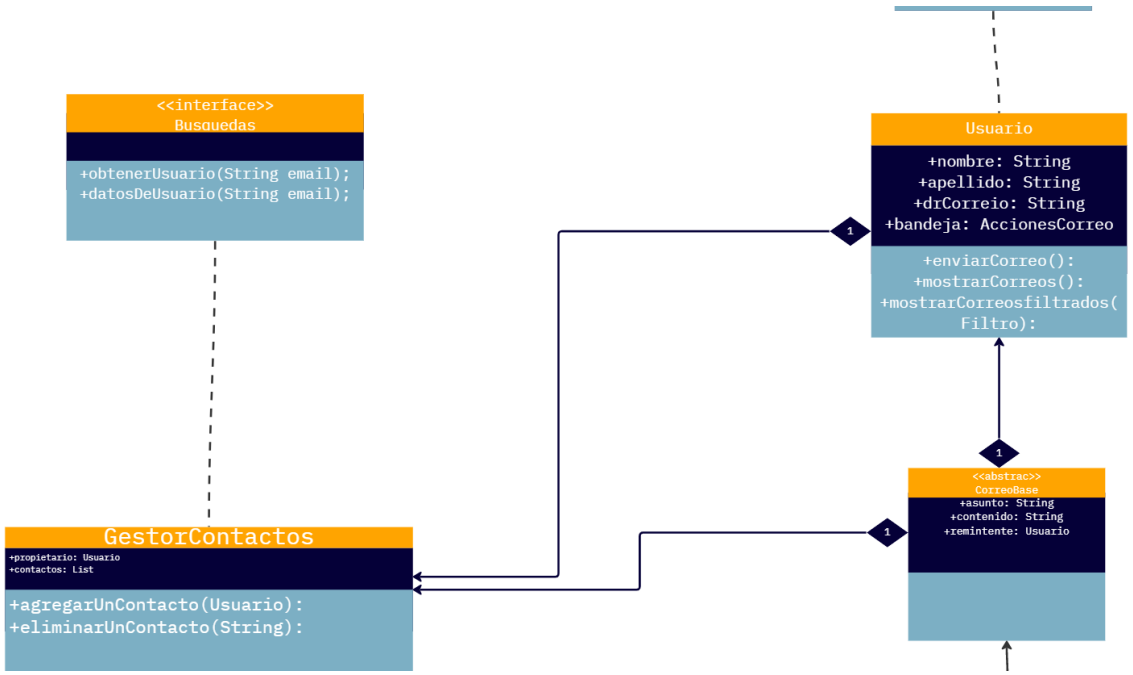
Presentacion final

Recursos Utilizados

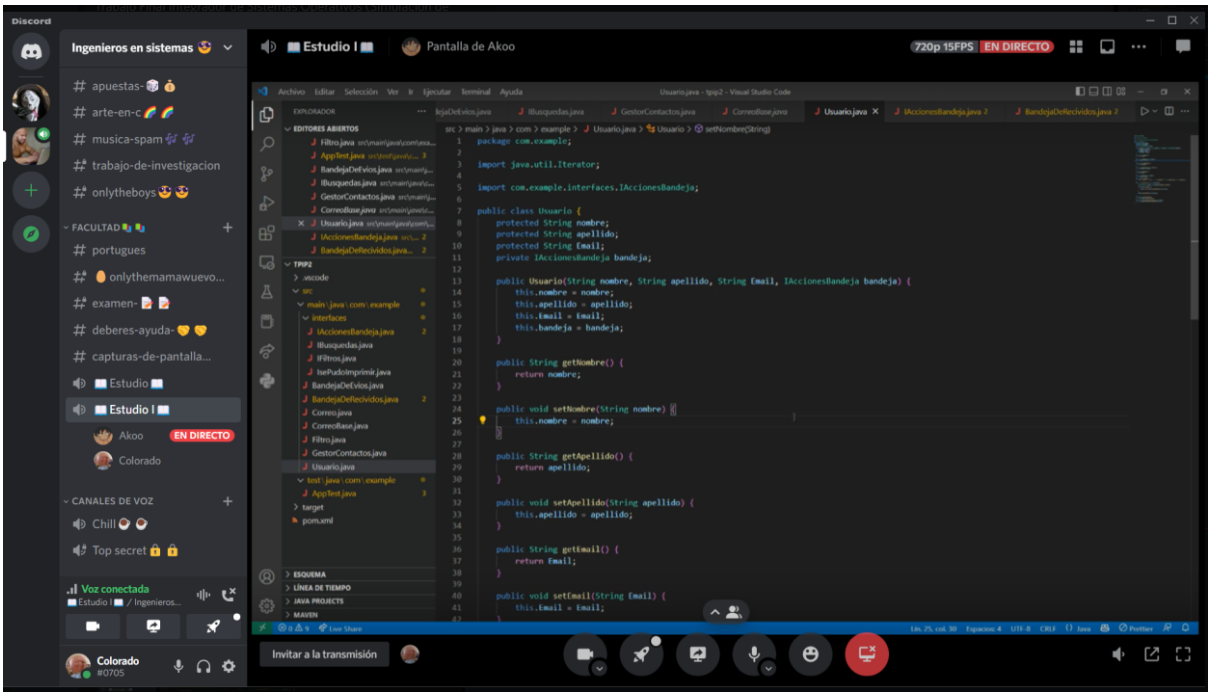
- **Desarrollo de la documentación:** Word, Drive y PDF.
- **Lenguaje Utilizado:** Java.
- **Comunicación:** Webex y Discord, WhatsApp, Slack.
- **Gestión de proyecto:** Discord.
- **Presentación:** PowerPoint.
- **Normas APA :** Séptima Edición.

Imágenes del proceso de desarrollo

Diseño del diagrama de clases en Miro



Al momento de trabajar sobre el código decidimos comunicarnos mediante discord.



Discord-dia 3:

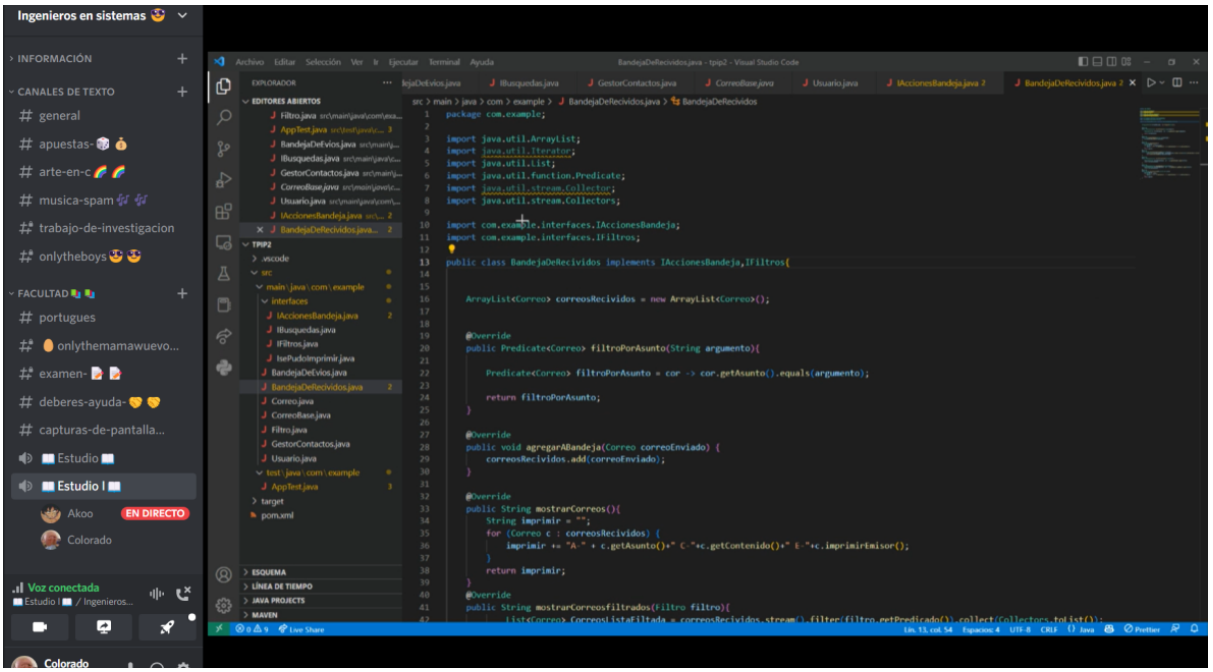
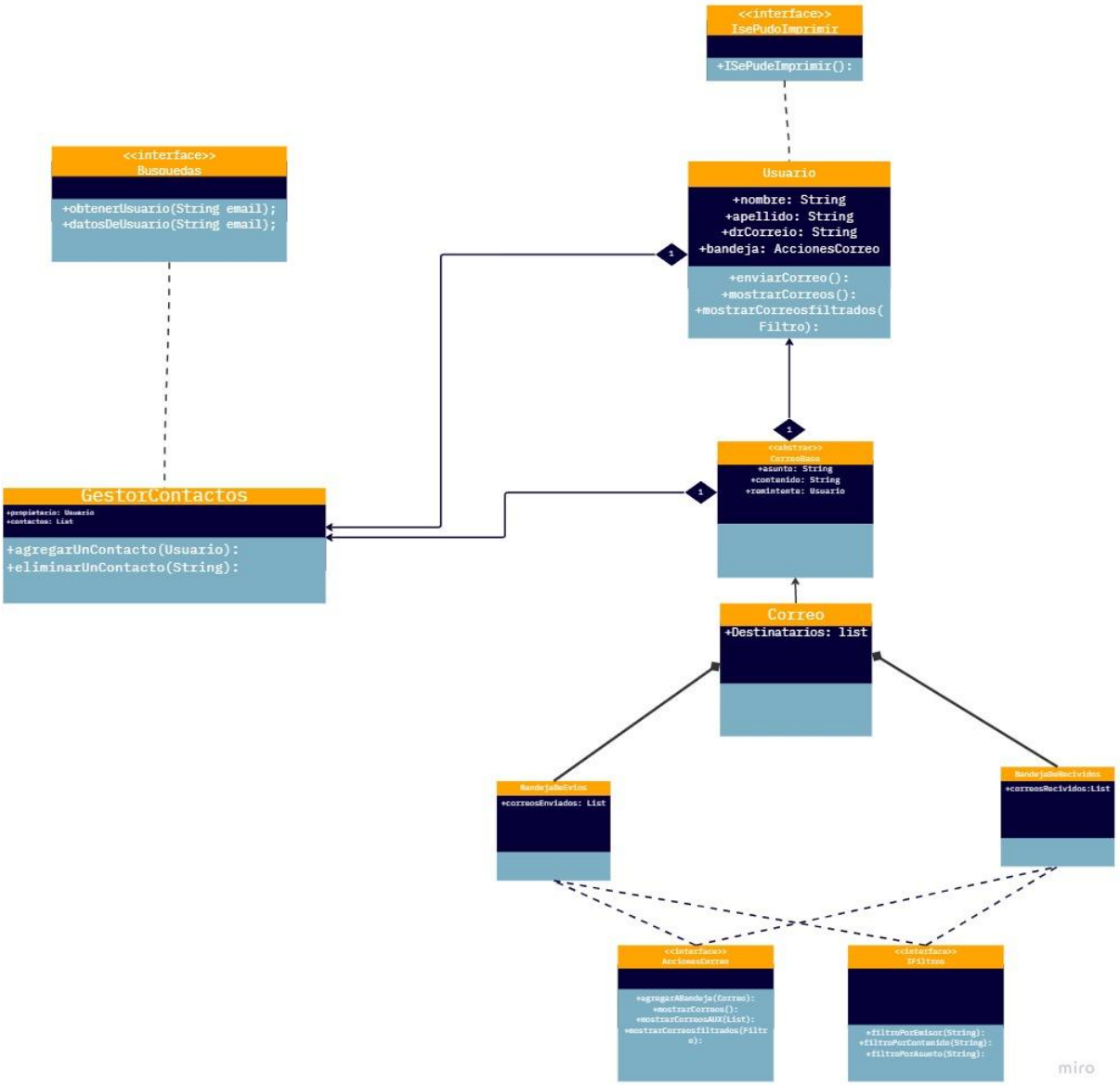


Diagrama de clases:



Propuesta de trabajos futuros

Para el futuro del trabajo proponemos implementar una interfaz gráfica al algoritmo y añadir nuevos métodos y funcionamientos a las clases así como expandirlo.

Conclusión individual

- **Suarez, Joaquin**

  Personalmente me resultó un desafío bastante bueno, ya que me forzó a imaginarme como podría ser el algoritmo, buscar y recopilar información para la solución, implementar los principios “SOLID” para que el código sea de calidad.Todo

esto me ayudó a mejorar mucho mis conocimientos sobre la programación orientada a objetos.

- **Briend, Andres**

  El proceso del trabajo me gusto mucho, ya que fue desafiante y me permitió sellar conocimientos durante el cursado de la materia. Por otra parte me resultó muy interesante e importante la implementación de los principios "SOLID" ya que este permite poder tener un codigo de muy buena calidad pero más allá de eso me permitió obtener una facilidad para poder interpretar el código, brindando legilibidad a la hora de ver y entender lo que implantaba mi compañero de trabajo.

Conclusión grupal

Una vez finalizado el trabajo llegamos a la conclusión de que la idea de trabajar en grupo facilita en gran medida el desarrollo del integrador, ayudando a dividir las tareas realizándose de una manera óptima, implementando el "concepto" del **"divide y vencerás"** como se nos fue enseñado a lo largo de estos años.

Algo a tener en cuenta fueron las diferentes ideas y caminos que quisimos tomar a la hora de desarrollar el diagrama de clases y el algoritmo. Pero esto lo superamos como equipo planteando nuestras ideas y llegando a un acuerdo de que sería lo más lógico y claro. De esta forma desarrollando el trabajo integrador con un algoritmo de calidad y muy escalable.

Links:

Diagrama de clases en miro. LINK.

Código en GITHUB. LINK.

Presentación en Power. LINK

Bibliografía

https://devexperto.com/principios-solid/

miro.com/app/board