URL Repositorio: https://bitbucket.org/DPiscoL/poo5\_proy2\_pisco\_campana\_santillan/src/master/

Fecha: 04/09/2020

Integrantes:

Andrés Campaña

Dustin Pisco

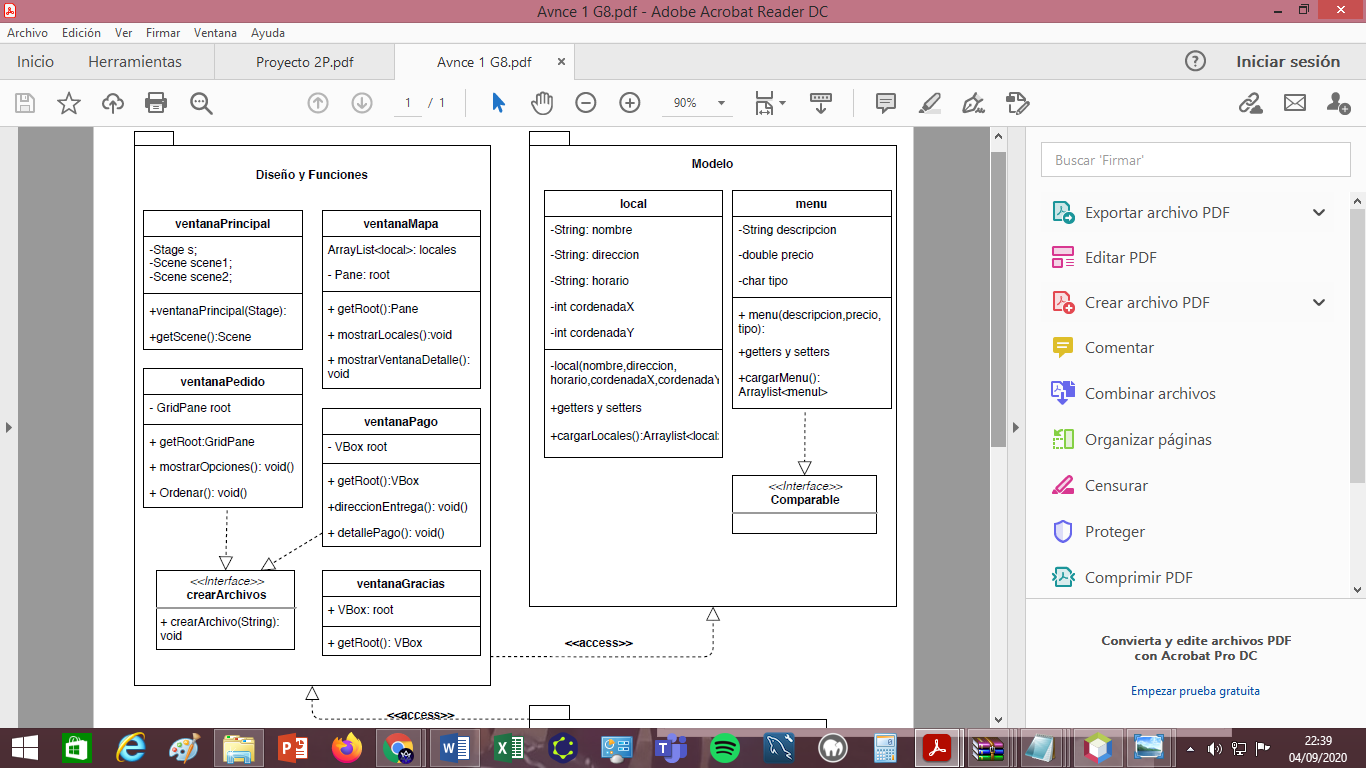
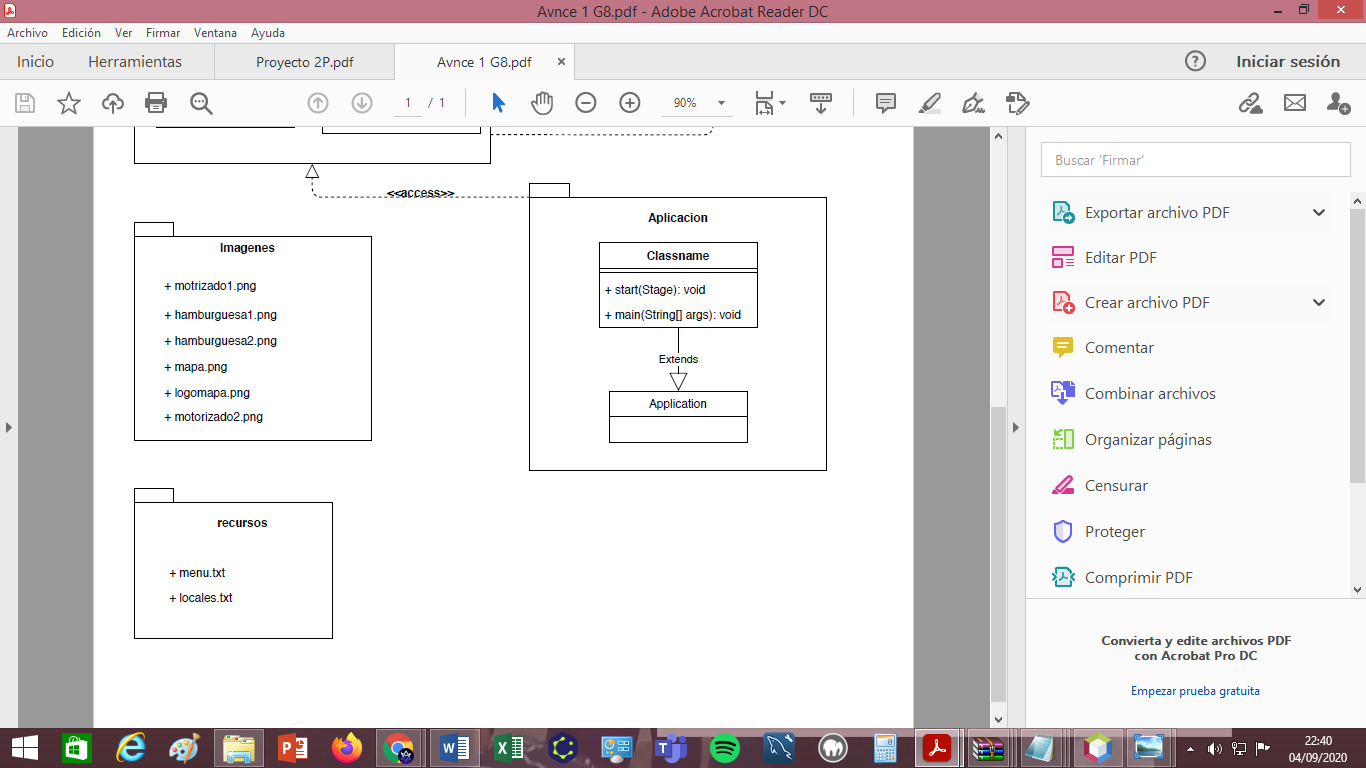
Karen Santillan

Reporte – proyecto SEGUNDO parcial-par 3

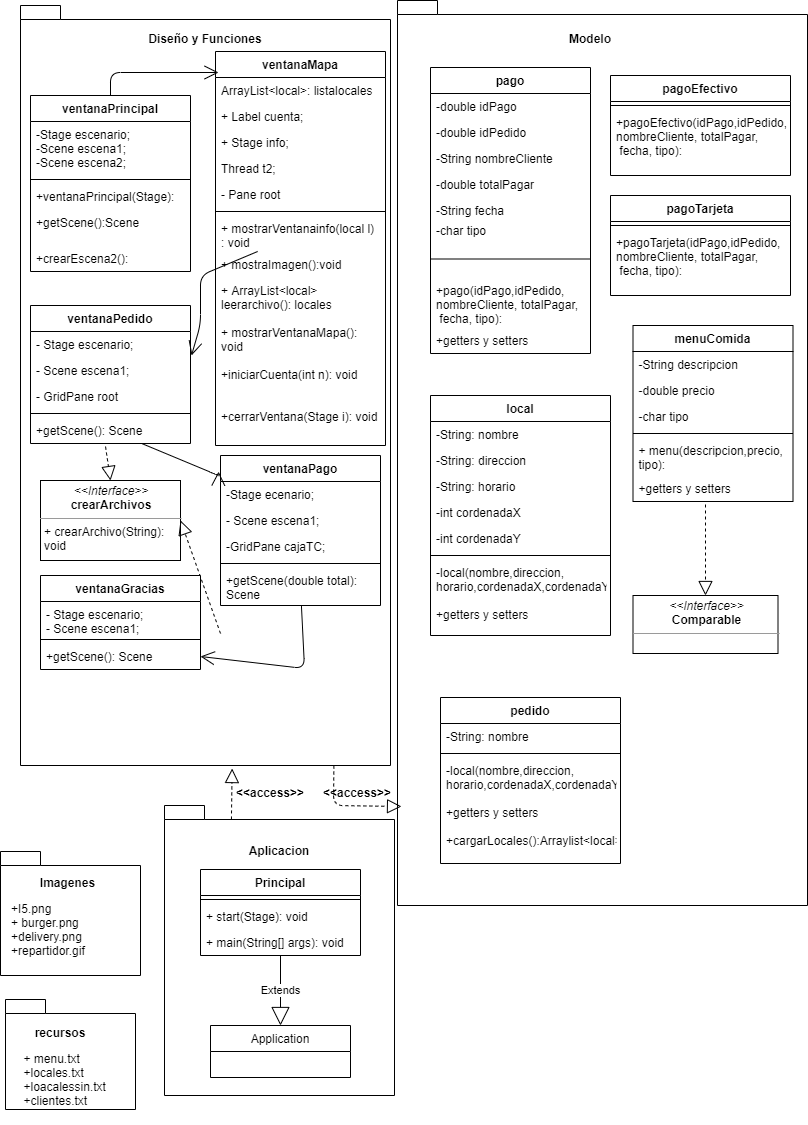
PROGRAMACIÓN ORIENTADA A OBJETOS TÉRMINO II 2019-2020

# Diagrama de clases

En esta sección deberán incluir el diagrama de clases que elaboraron como primera versión de su proyecto.



SEGUNDA VERSION DEL UML



# Tareas

En esta sección incluirán el detalle de las tareas que se asignó a cada estudiante.

Estudiante (Andres Campaña):

1. Creación de clases y paquetes para el proyecto
2. Agregar archivo de texto e imagen, programar eventos para botón de la ventana principal
3. Crear eventos para mostrar ventanas

Estudiante (Dustin Pisco):

1. Diseño clase principal
2. Diseño de ventana (excepto mapa)
3. Creacion del proyecto

Estudiante (Karen Santilllan):

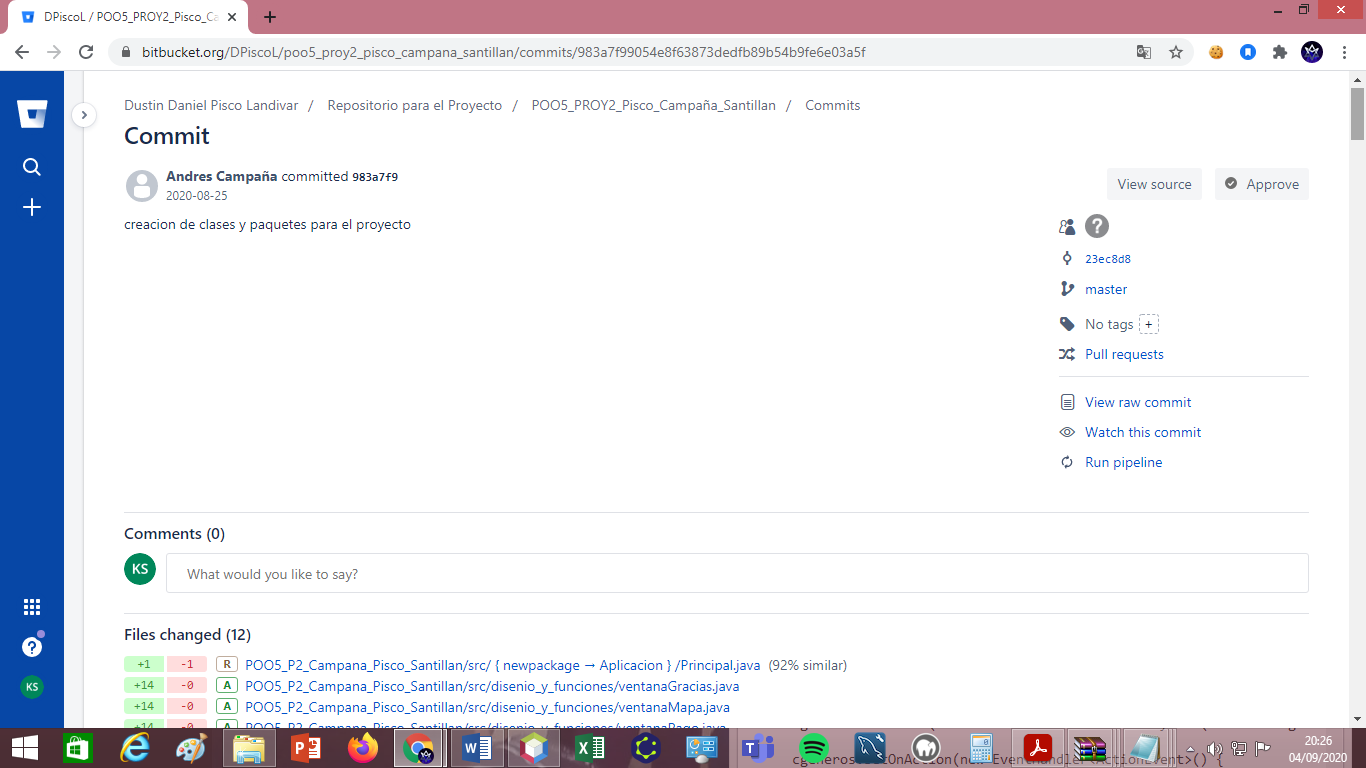
1. Programar eventos para botones efectivo y tarjeta de credito
2. Programar eventos para botones continuar y limpiar
3. Validación de usuario y declaración de eventos del boton y encuentra.

# Evidencias de Tareas

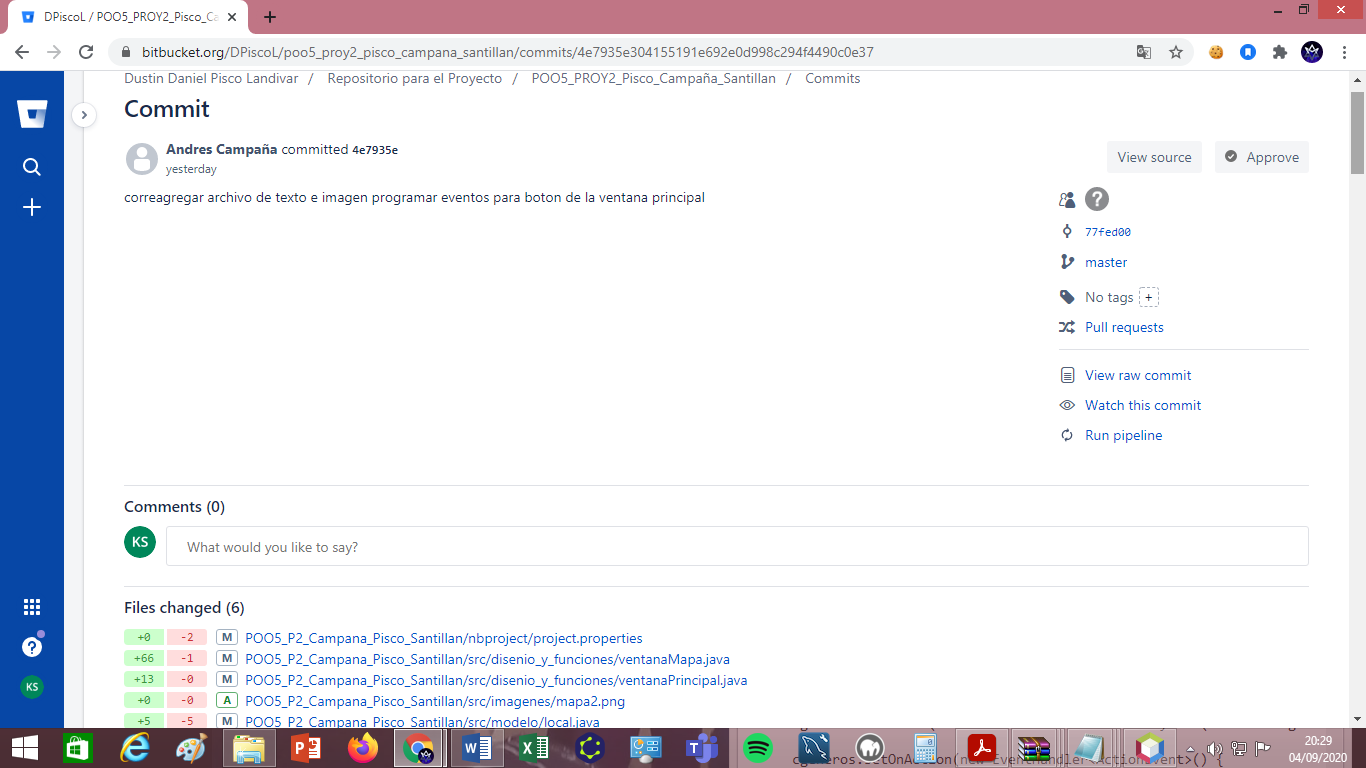
En esta sección incluirán screenshots de la página de Bitbucket donde consten los commits realizados. Además, por cada estudiante deben elegir un par de commits y mostrar el detalle. Esto debería concordar con las tareas definidas para cada uno. Ejemplo:

Estudiante (Andrés Campaña):

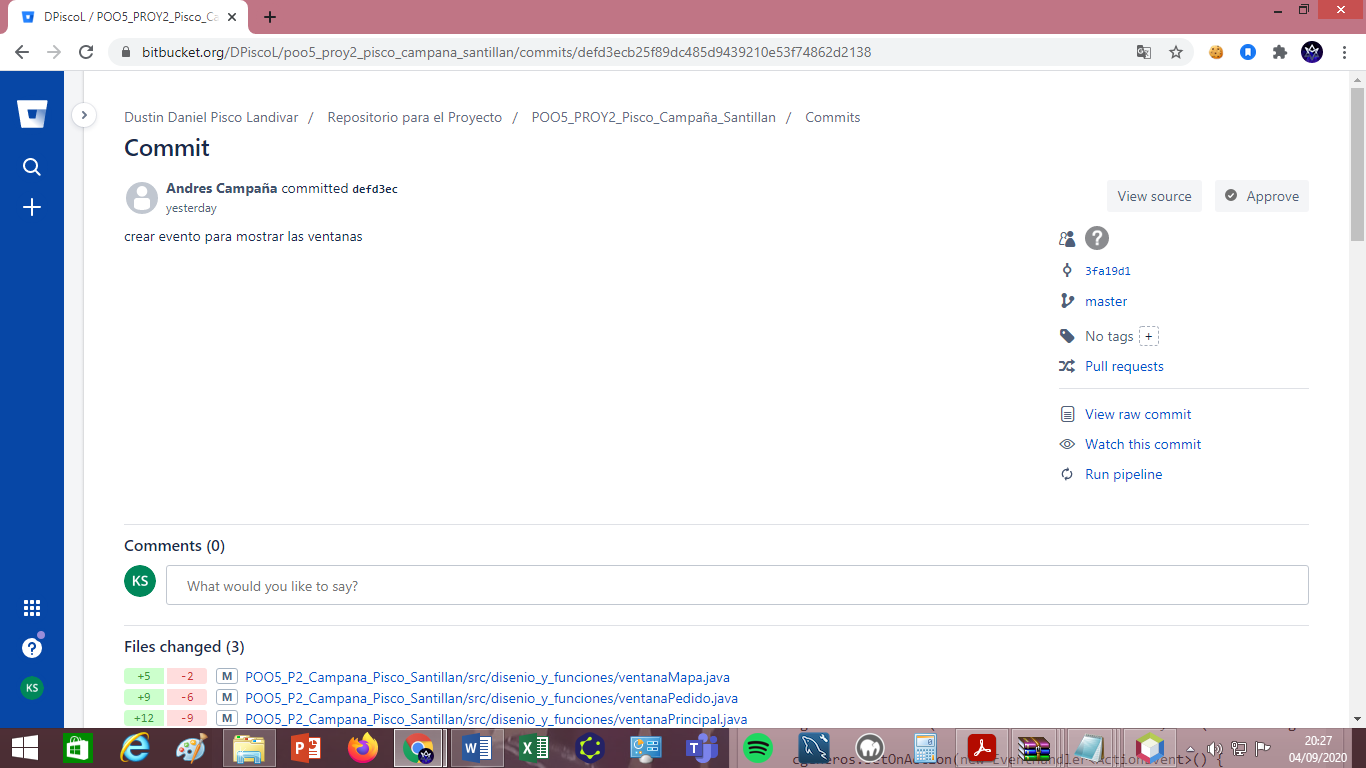
Commit 1



Commit 2



Commit 3

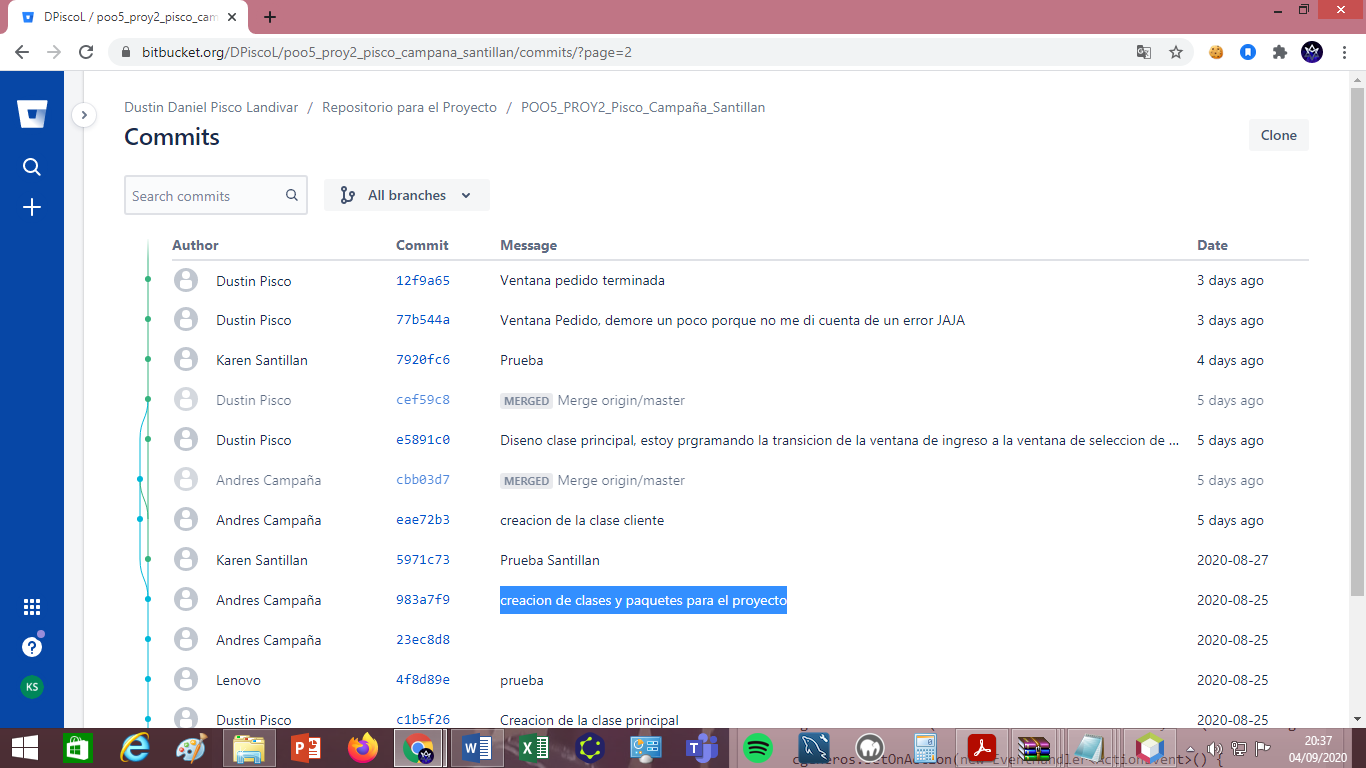


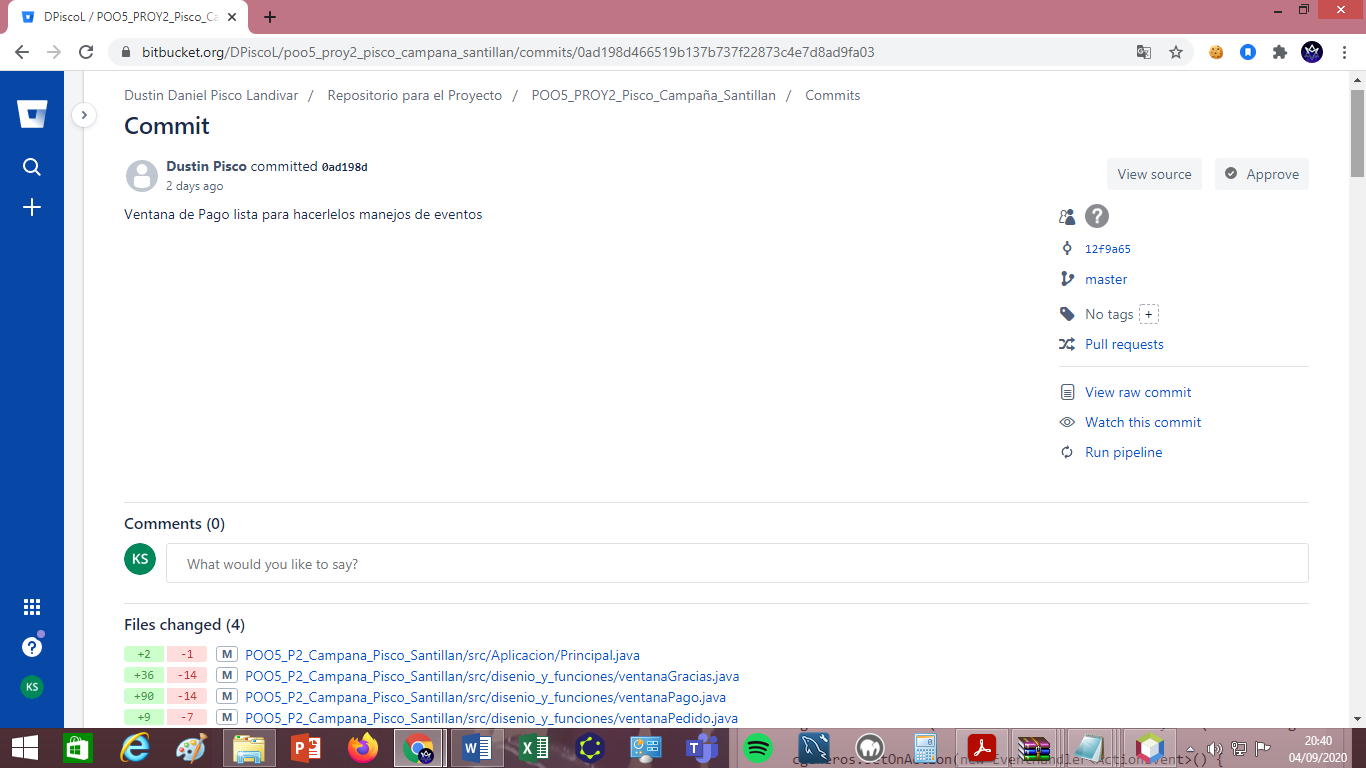
Estudiante (Dustin Pisco):

Commit 1

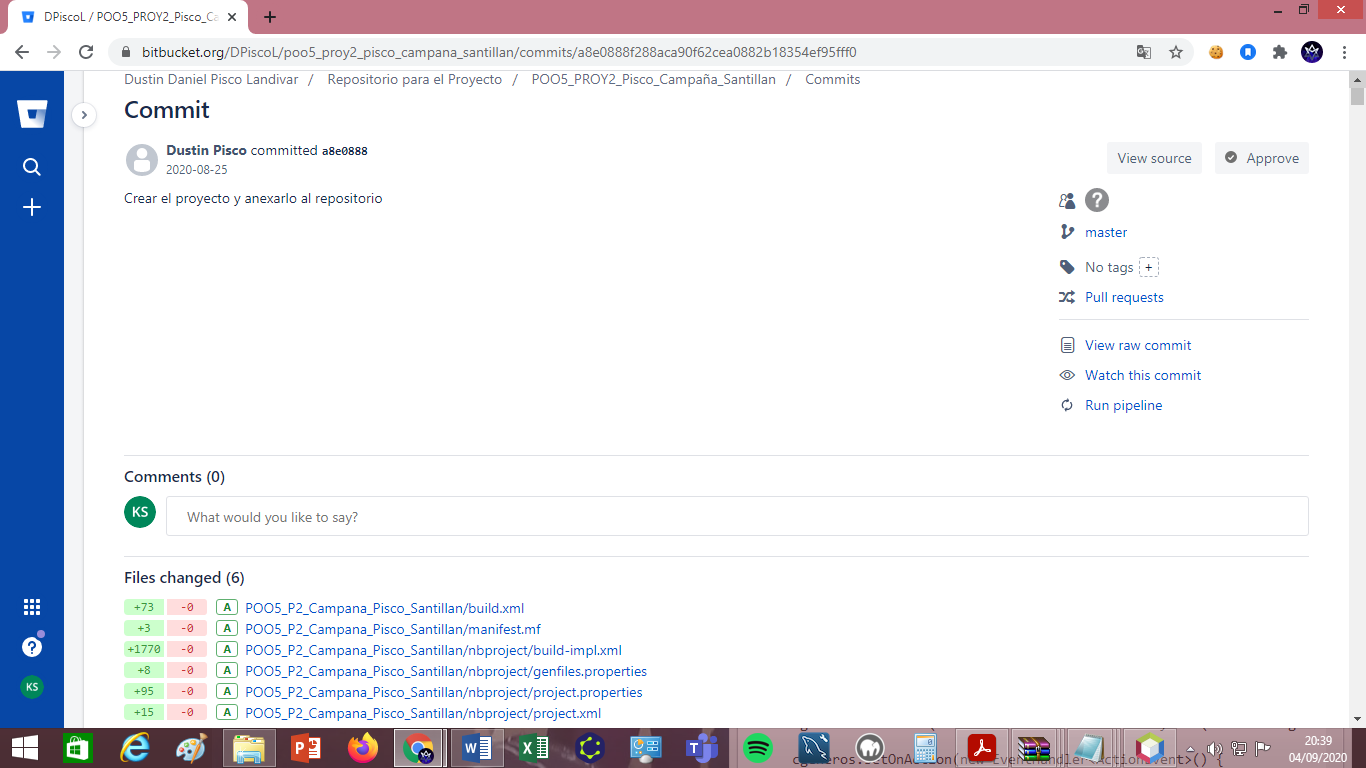


Commit 2



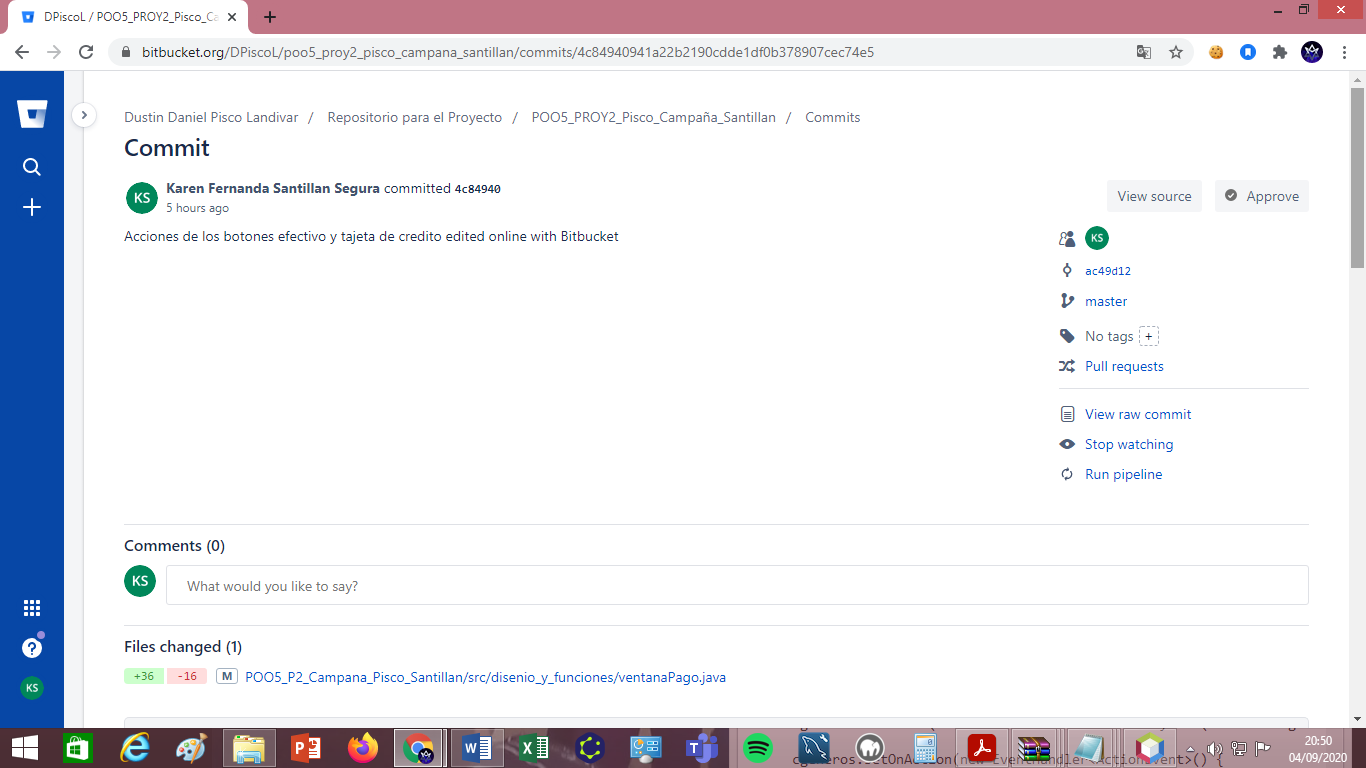


Commit 3

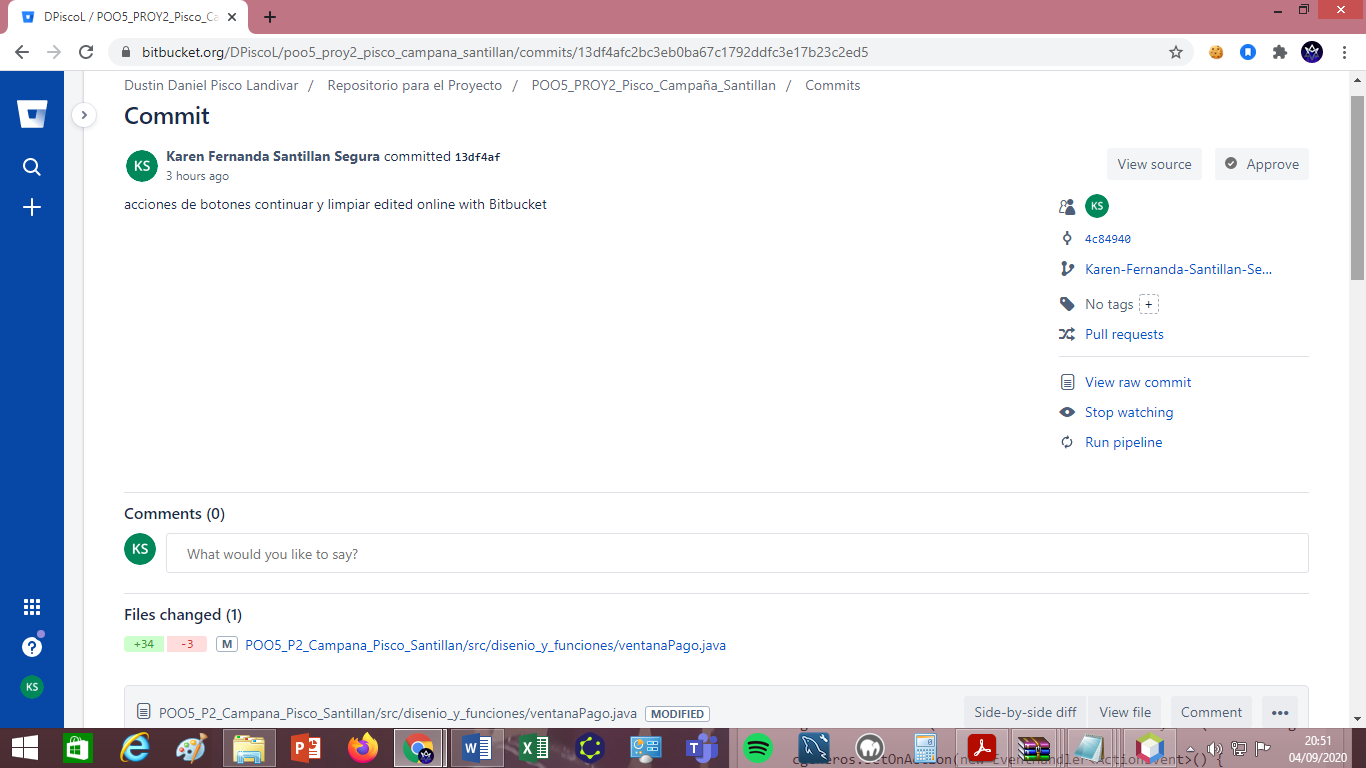


Estudiante (Karen Santillan):

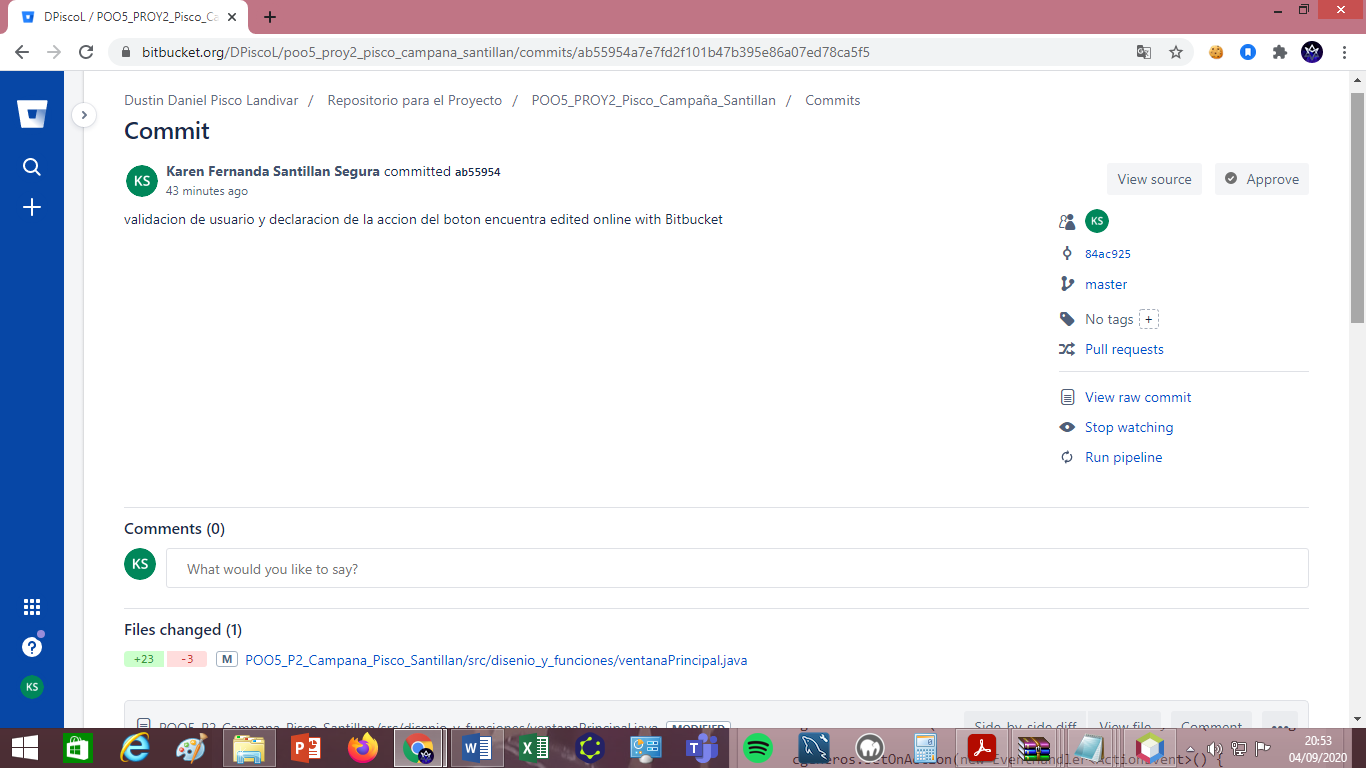
Commit 1



Commit 2



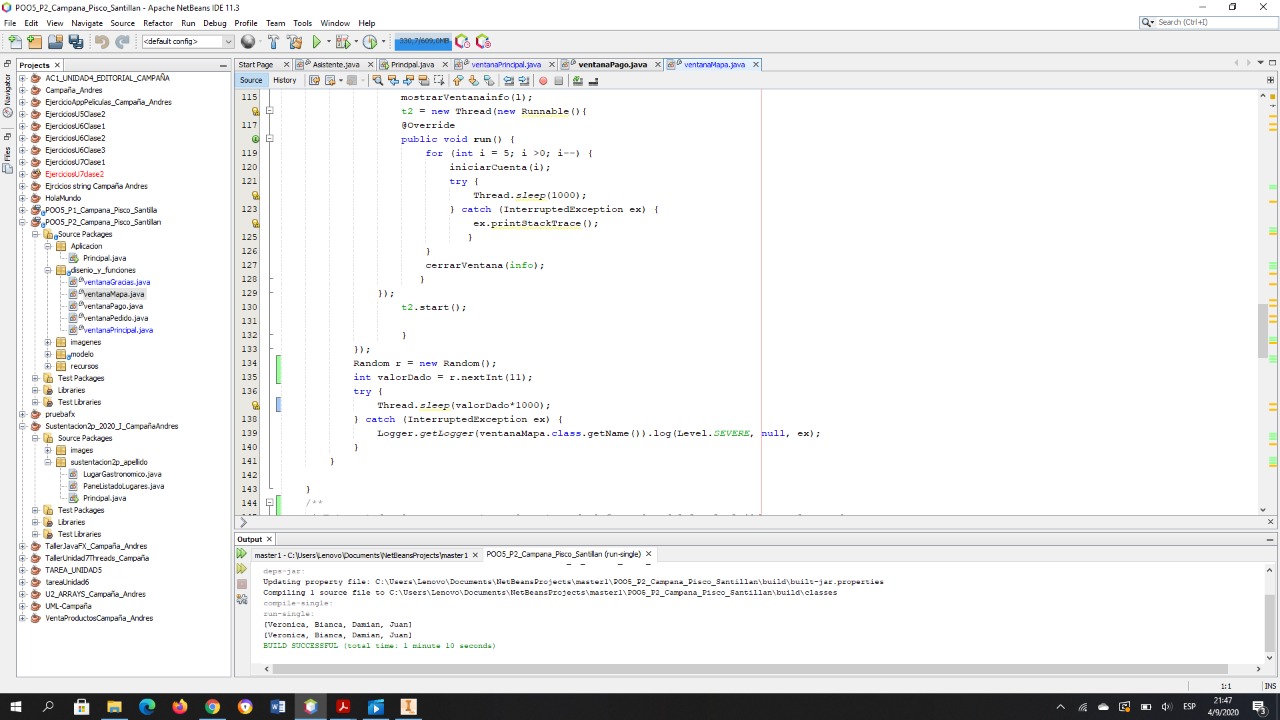
Commit 3

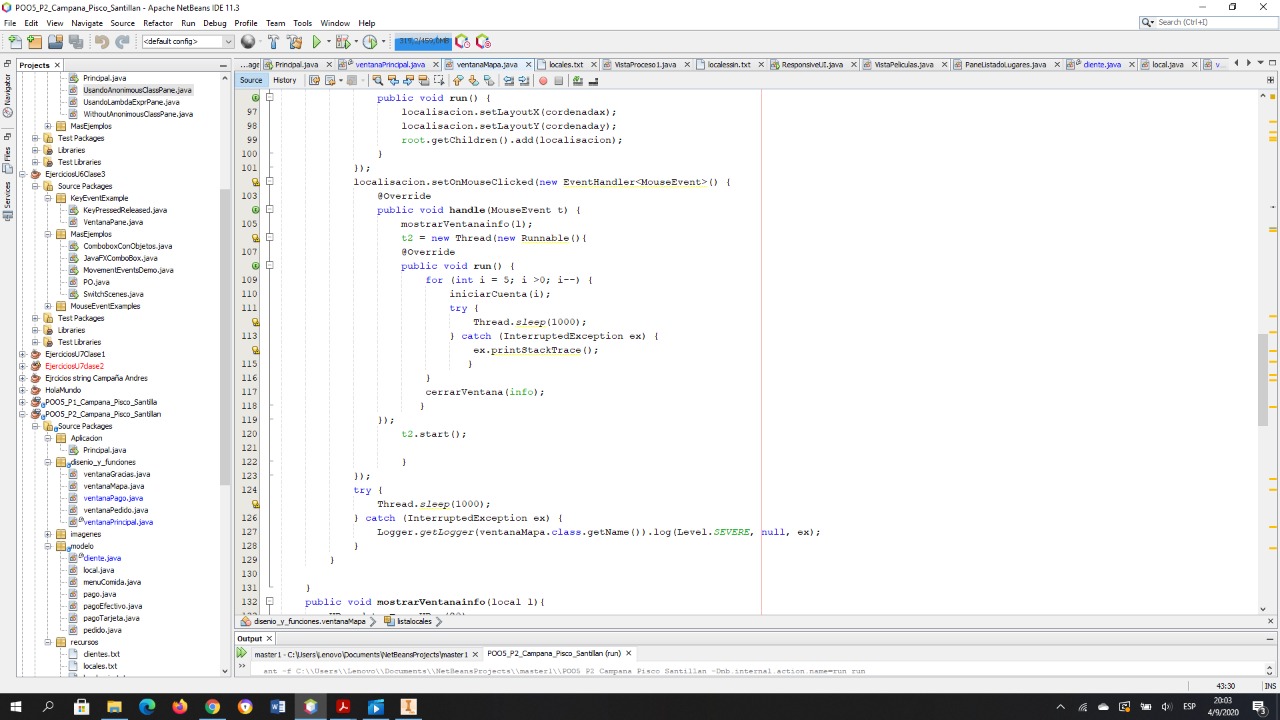


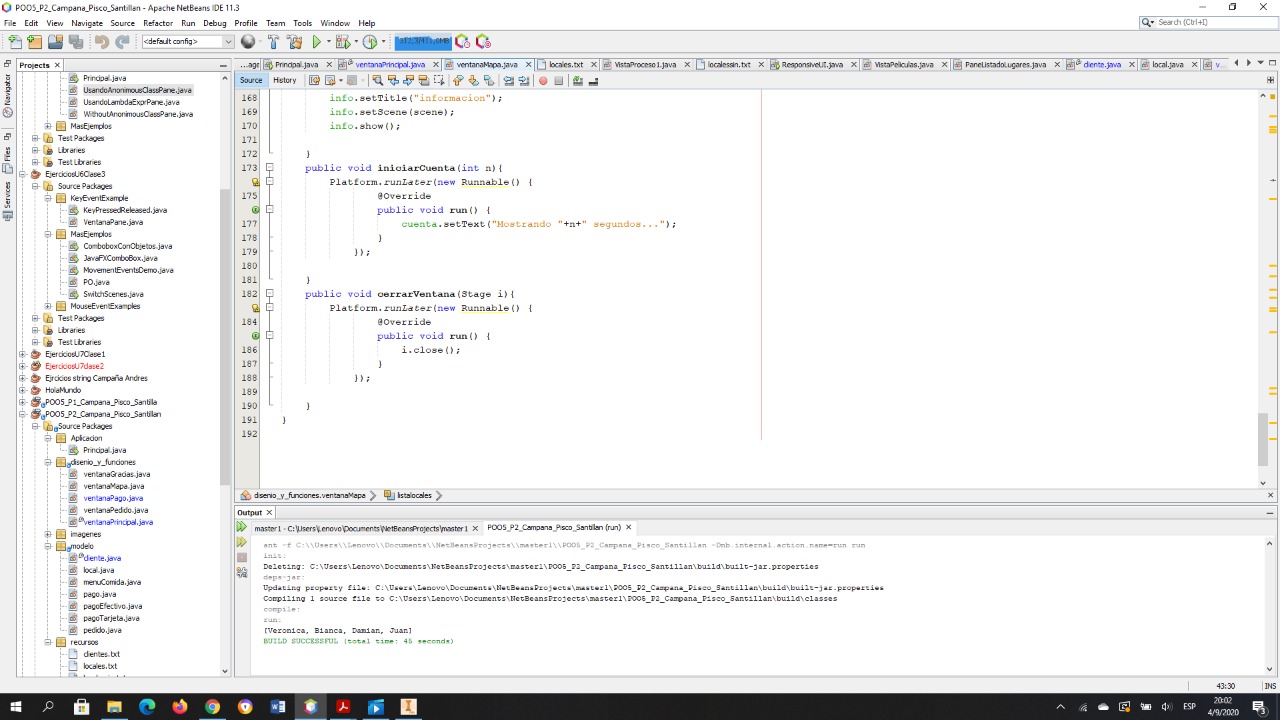
# Identificación de teoría aplicada en programación orientada a objetos.

En esta sección deberán incluir el detalle de los bloques de código de su proyecto donde identifican cada pilar de este paradigma: Abstracción, Encapsulamiento, Herencia y Polimorfismo. Además, deberán mostrar donde usan la interface Comparable y una descripción de porqué se ha decidido aplicar el usar de cada uno de estos conceptos en sus proyectos. Si no incluyen la explicación adecuada, este punto no será válido. Ejemplo:

## **Threads**





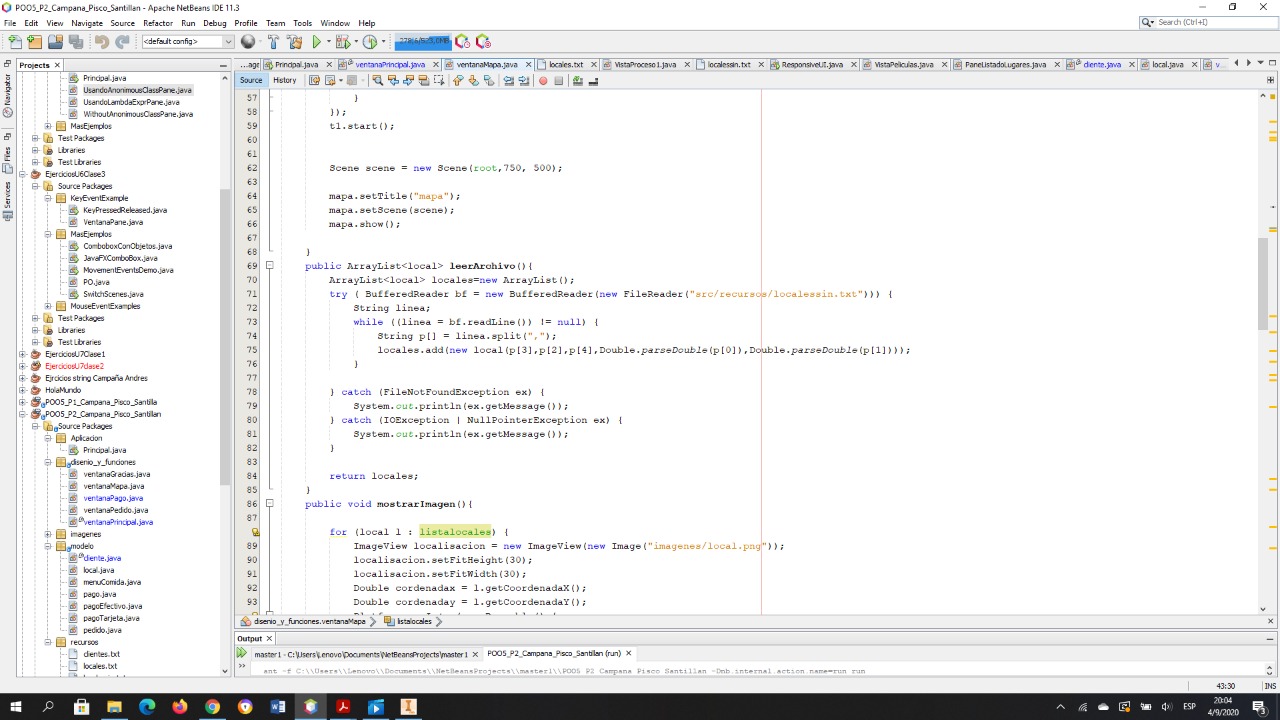


Explicación

En la primera imagen se logra establecer un tiempo para la ejecución de una acción a través de un Thread, en este caso este nos permite mostrar en la pantalla el mapa con las ubicaciones de los locales revelando una a una dichas ubicaciones en intervalos de tiempo.

De eta manera logramos realizar diversas acciones de forma simultánea brindando comodidad al usuario.

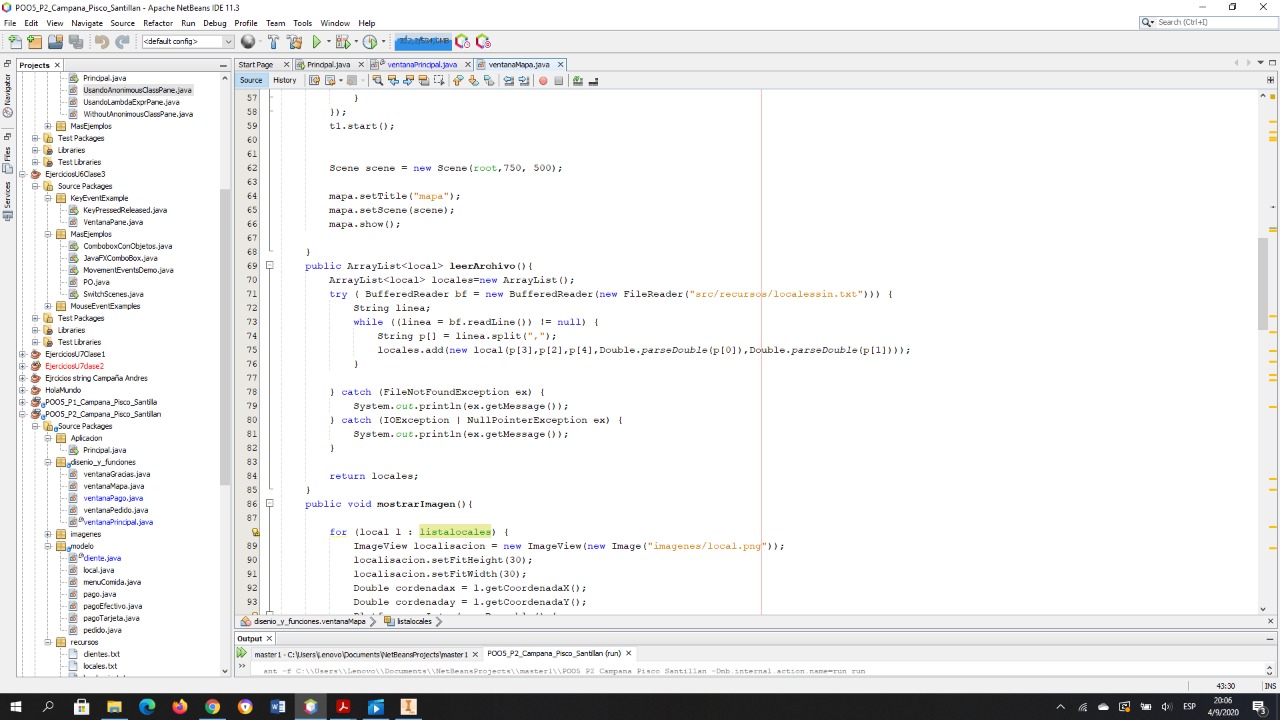
## **Manejo de Excepciones**



Explicación

Cuando se lee un archivo pueden existir problemas o errores, tales como: que la url donde se encuentra el archivo sea errónea, errores de escritura dentro del archivo, etc. Estas excepciones son errores que pueden surgir en tiempo de ejecución o compilación y que se dan en la mayoría de casos por errores de sintaxis también pueden ocurrir en los threads también o cuando un campo ingresado por el usuario puede ser null. Debido a que estos errores pueden hacer que el programa no se ejecute de manera adecuada y no brinden una explicación entendible para el usuario, es necesaria manejar estas excepciones tratando de evitarlas, y en el caso de no lograrlo, mostrar un mensaje más “amigable” al usuario.

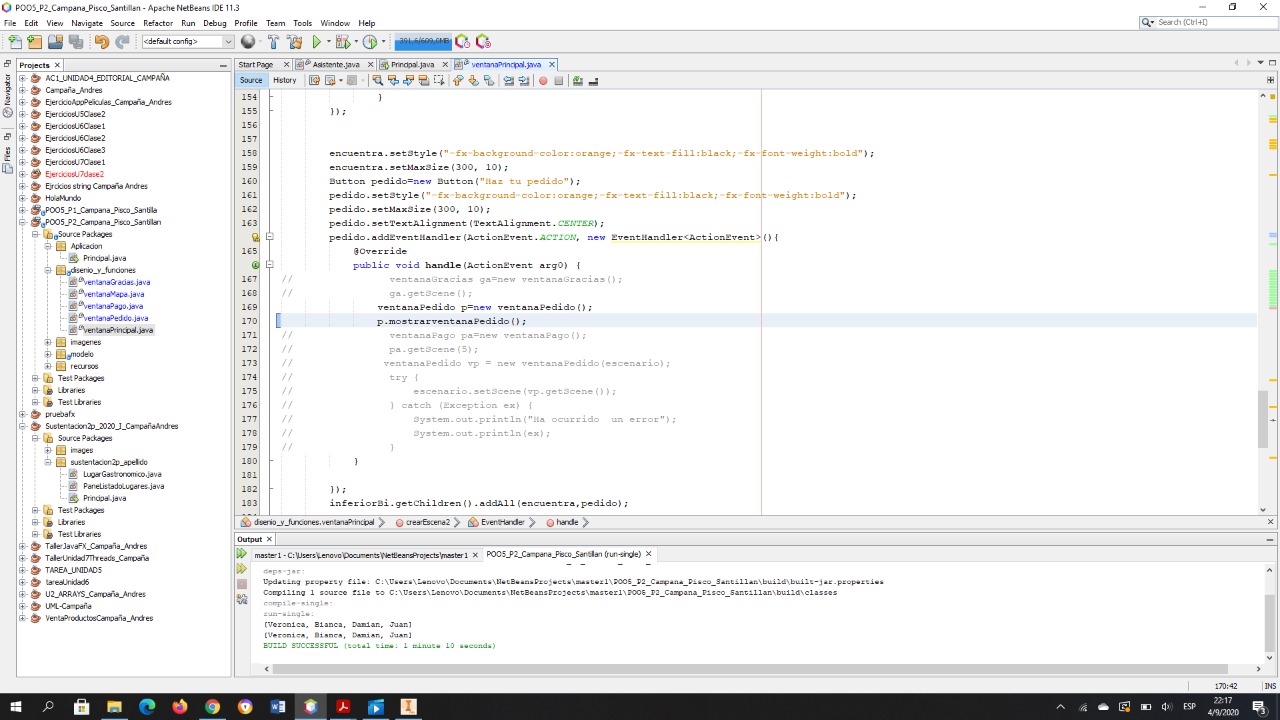
## **Manejo de Archivos**

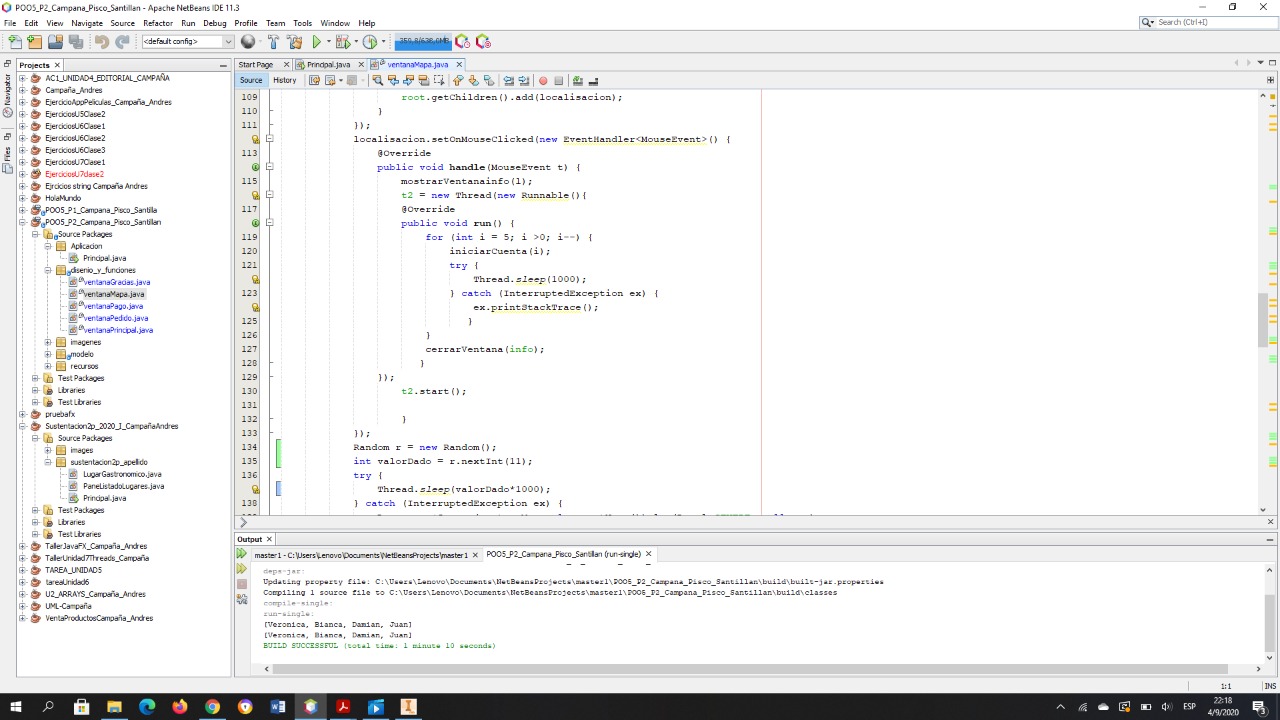


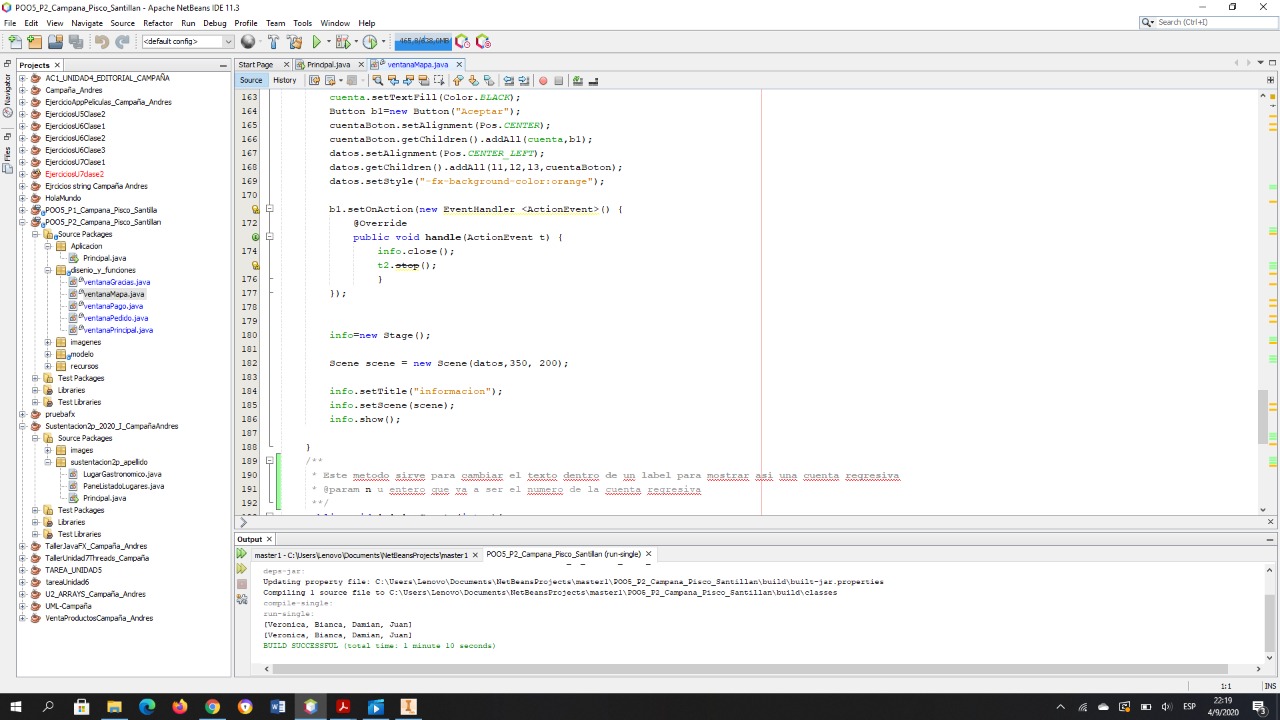
Explicación

Podemos ver cómo usar métodos de la clase abstracta Reader para obtener una conexión entre un stream de bytes y un stream de caracteres, de esta menra se logrará leer o escribir un archivo de datos ya sea de caracteres( texto,character based) o archivos de bytes(o binarios,bytes based)

## **Controladores de eventos**





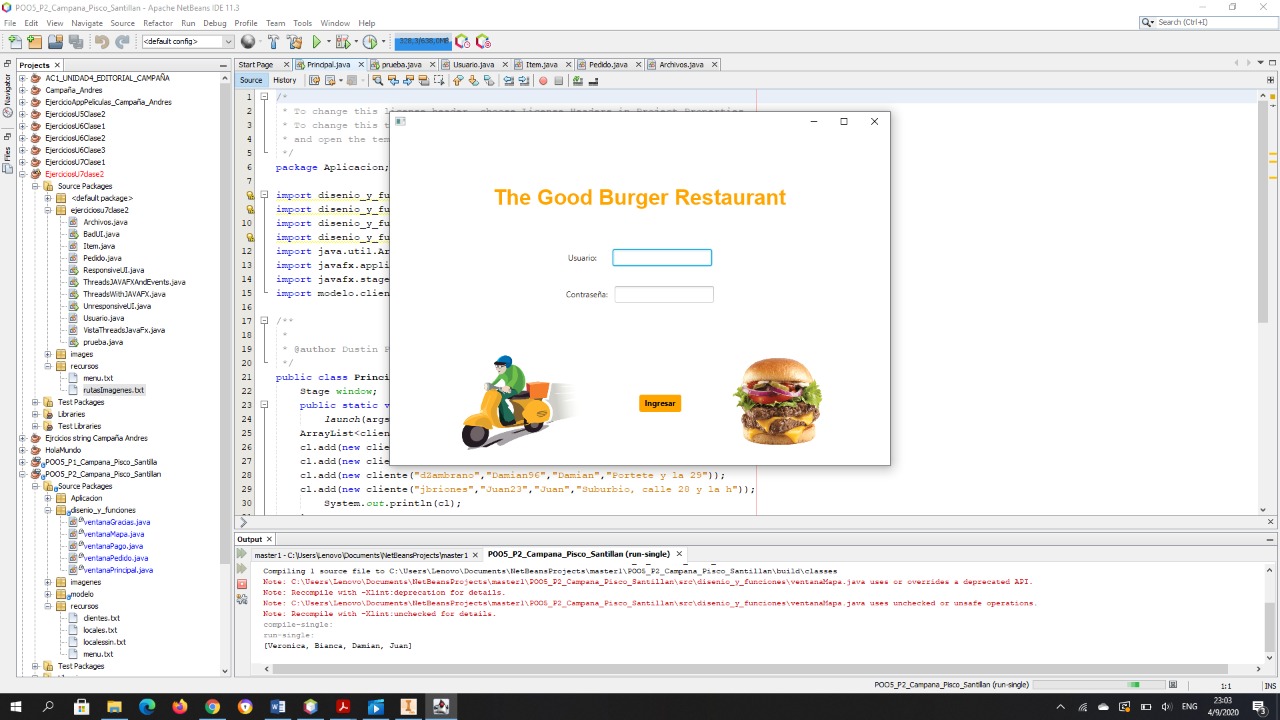


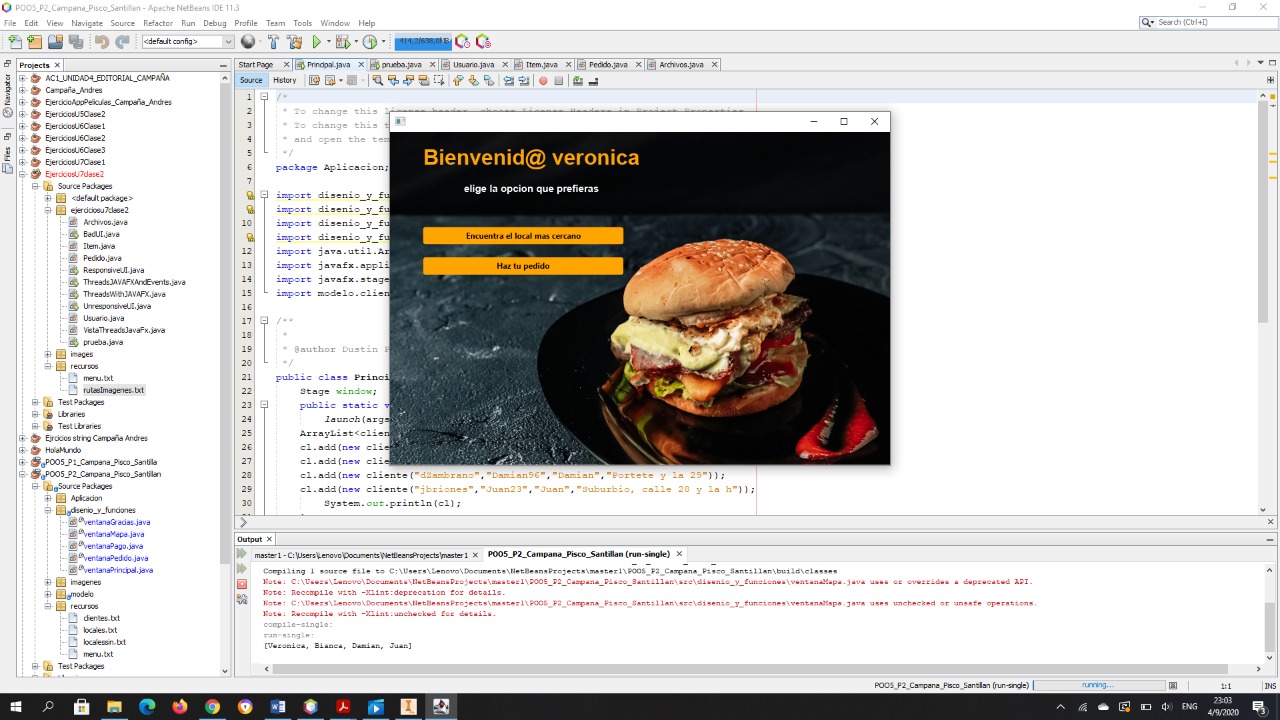
Explicación

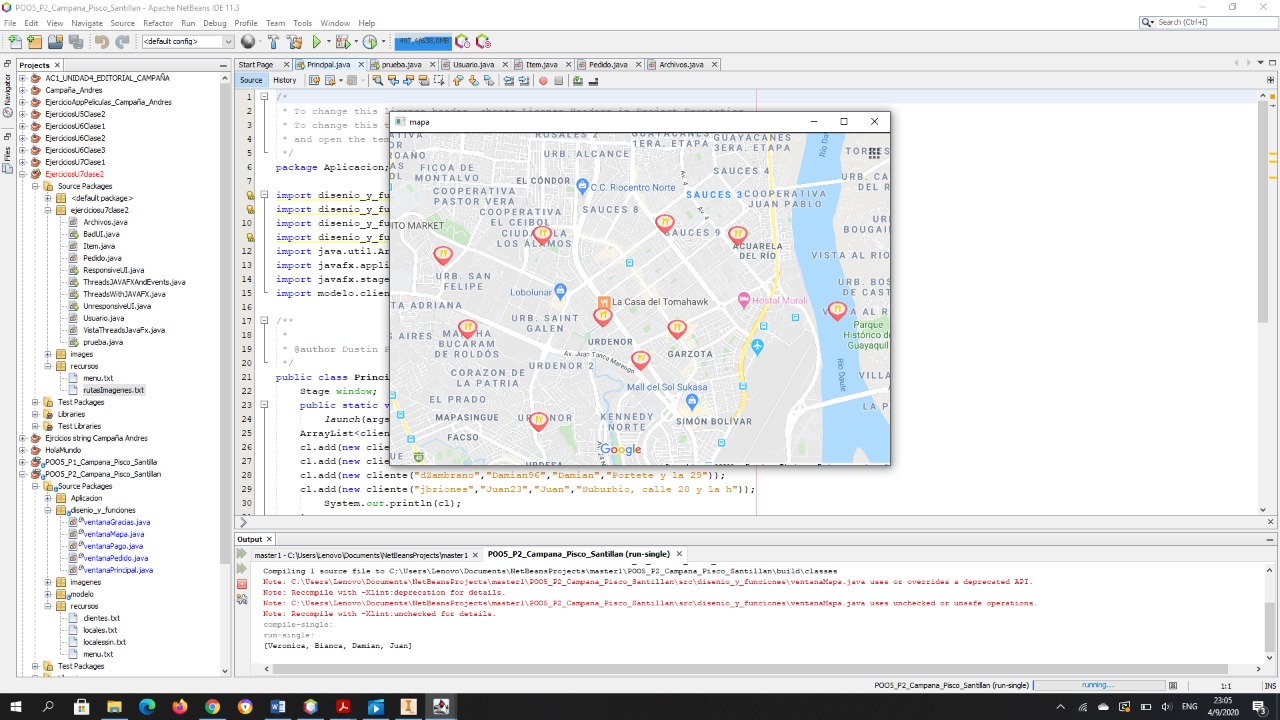
Gracias a las diversas herramientas que posee javaFX se pueden crear botones que ejecuten acciones, sin embargo estas acciones deben ser establecidas. Con los controladores de eventos logramos asignar las acciones que deben ejecutar los botones creados en las distintas ventanas, dichas acciones incluyen cambios de escenas, guardar información, cambios de ventanas, etc.

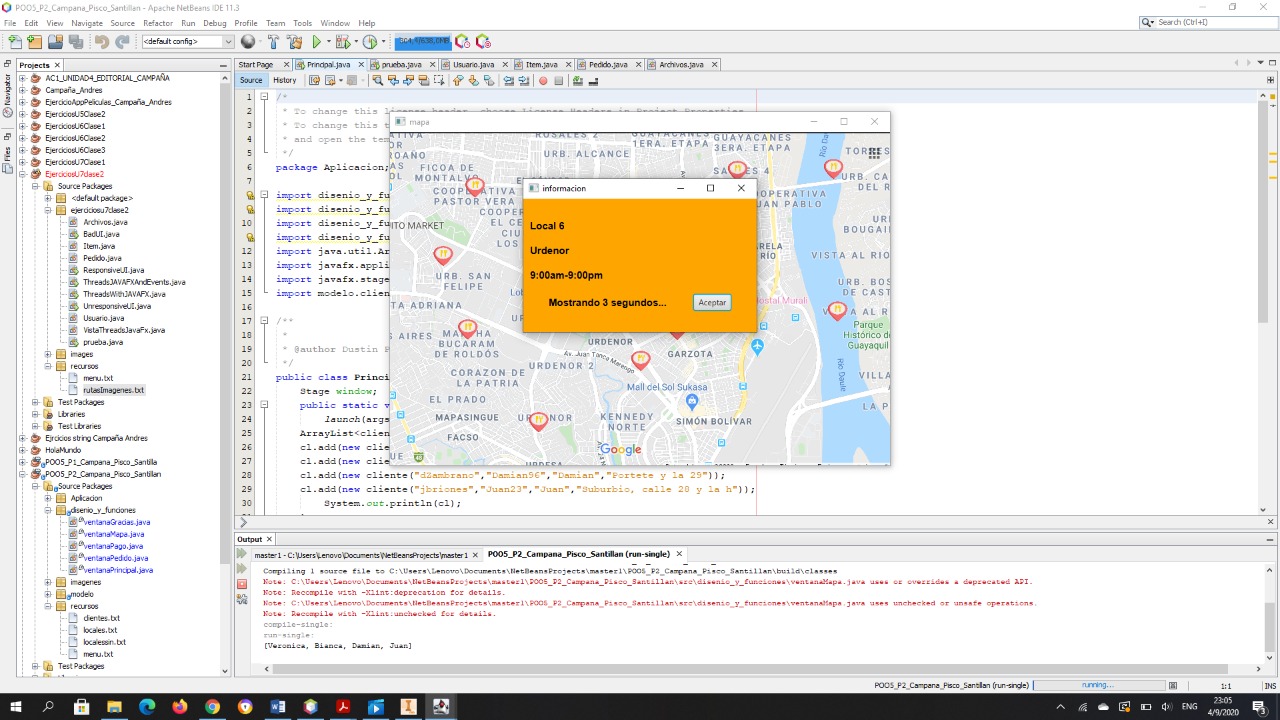
# Programa en ejecución

En esta sección deberán incluir screenshots de su proyecto en ejecución, con el objetivo de que se visualice la ejecución de este.









# JAVADOC

**Agregar la documentación JAVADOC como una carpeta adicional a este reporte**. Los métodos deben estar siempre comentados con el formato explicado en clase.