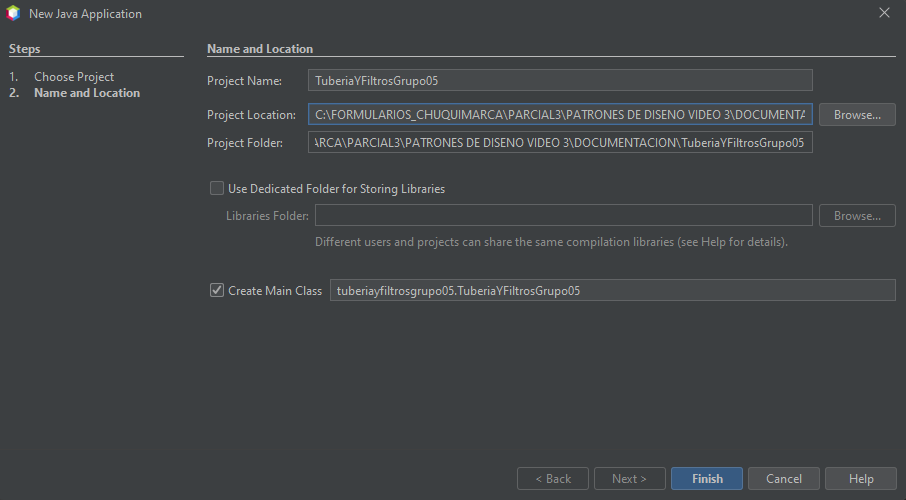
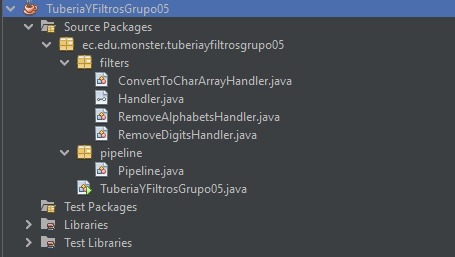
**CREACION DEL PROYECTO (TUBERIAS Y FILTROS)**

Supongamos que queremos pasar una cadena a una serie de etapas de filtrado y convertirla en una matriz de caracteres en la última etapa.



ESTRUCTURA INICIAL DEL PROYECTO



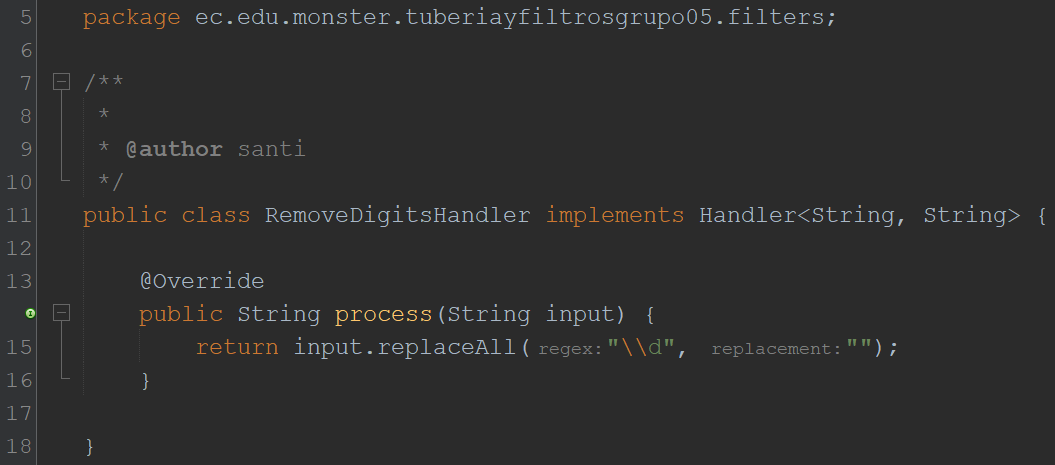
CODIFICACION PAQUTE FILTERS

Text

Description automatically generated

Text

Description automatically generated



Text

Description automatically generated

Codificación paquete pipeline

Text

Description automatically generated

Codificación main

Text

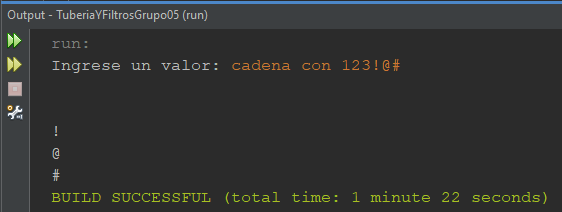
Description automatically generated

ESTRUCTURA FINAL DEL PROYECTO

Graphical user interface, text, application

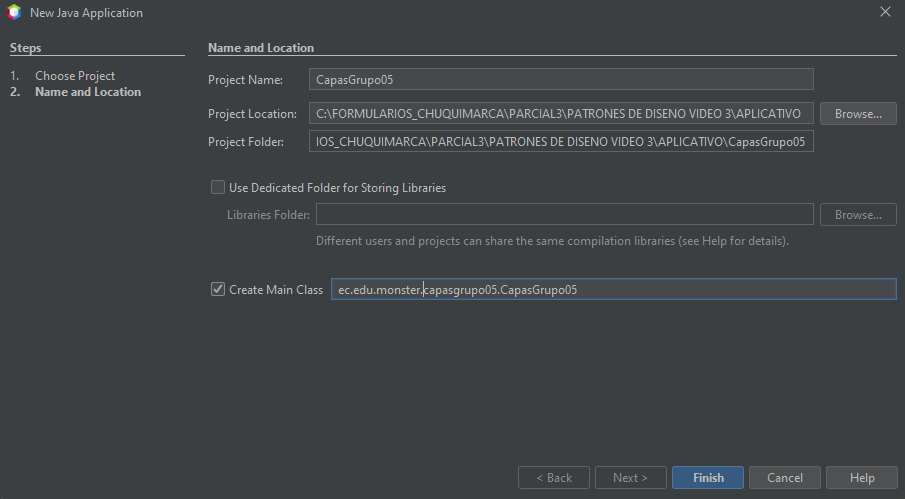
Description automatically generated

EJECUCION

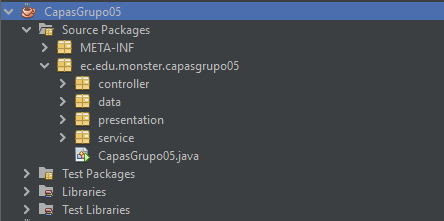


**CREACION DEL PROYECTO (CAPAS)**

Considere un sitio web que muestre pasteles decorados para bodas y cosas por el estilo. En lugar de que la página web acceda directamente a la base de datos, se basa en un servicio para entregar esta información. Luego, el servicio consulta la capa de datos para asimilar la información necesaria.



ESTRUCTURA INICIAL DEL PROYECTO

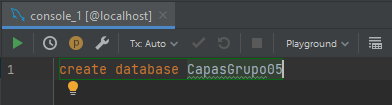


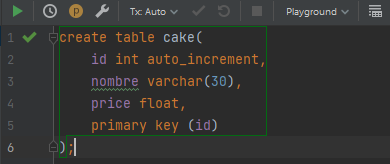
Capa de datos

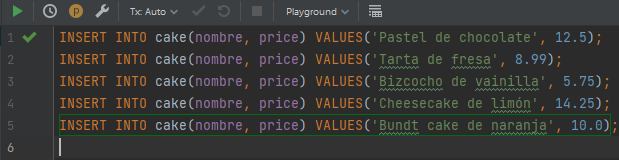
Usando un cliente de base de datos para mysql

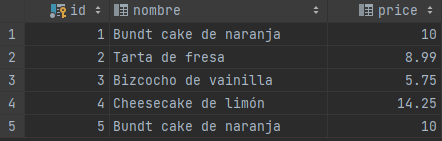
A screenshot of a computer

Description automatically generated









Graphical user interface, text

Description automatically generated

Graphical user interface, application

Description automatically generated

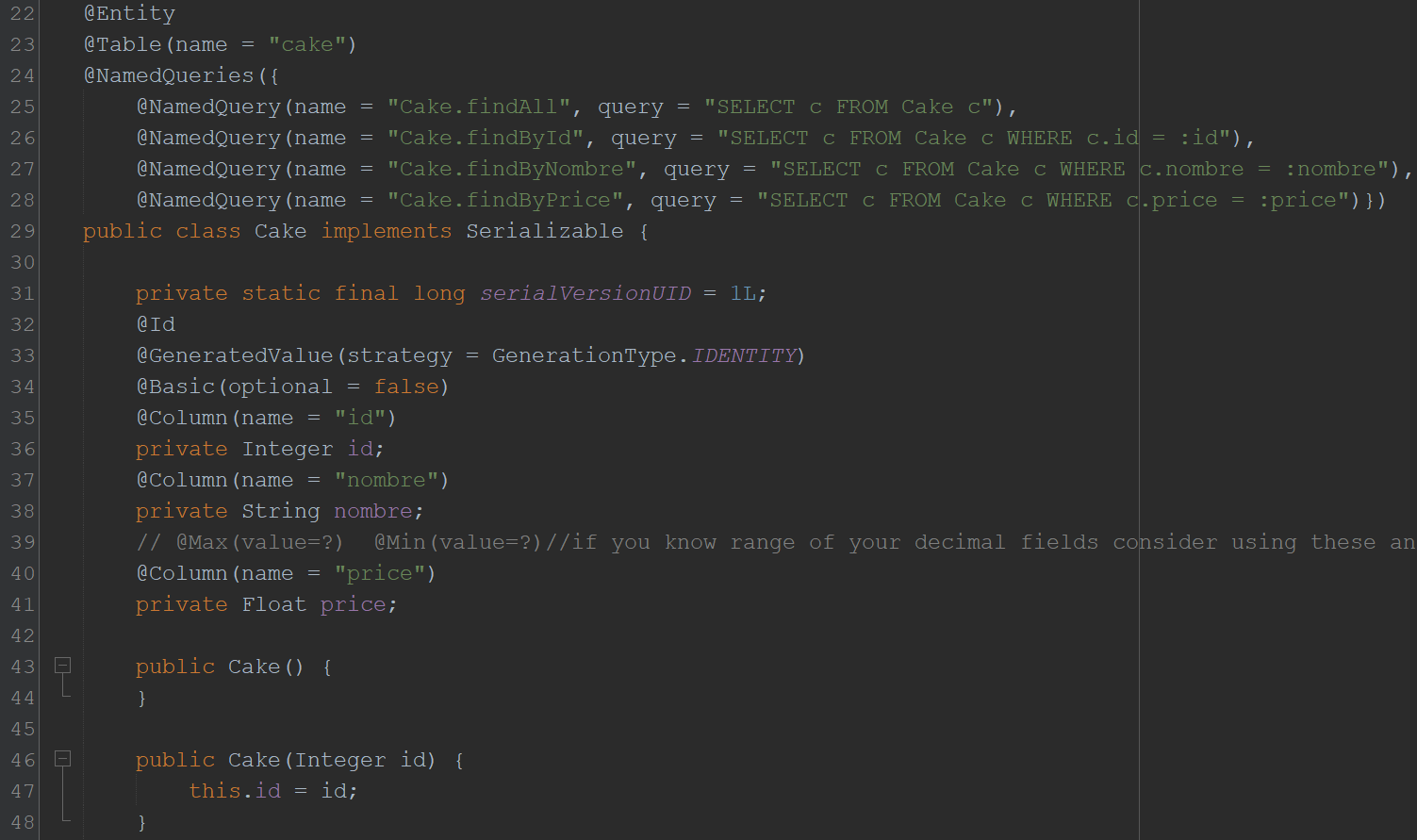
Graphical user interface

Description automatically generated

Text

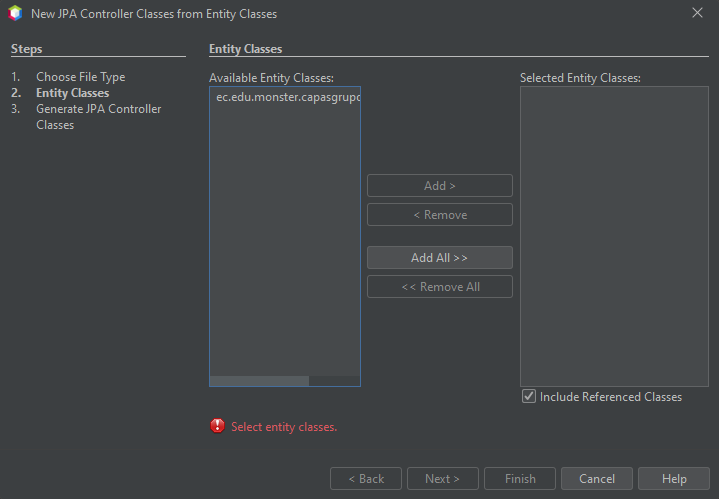
Description automatically generatedText

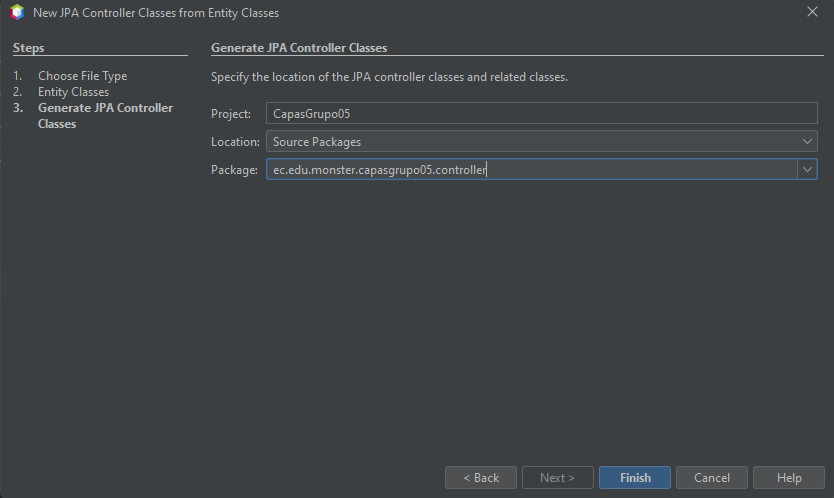
Description automatically generated

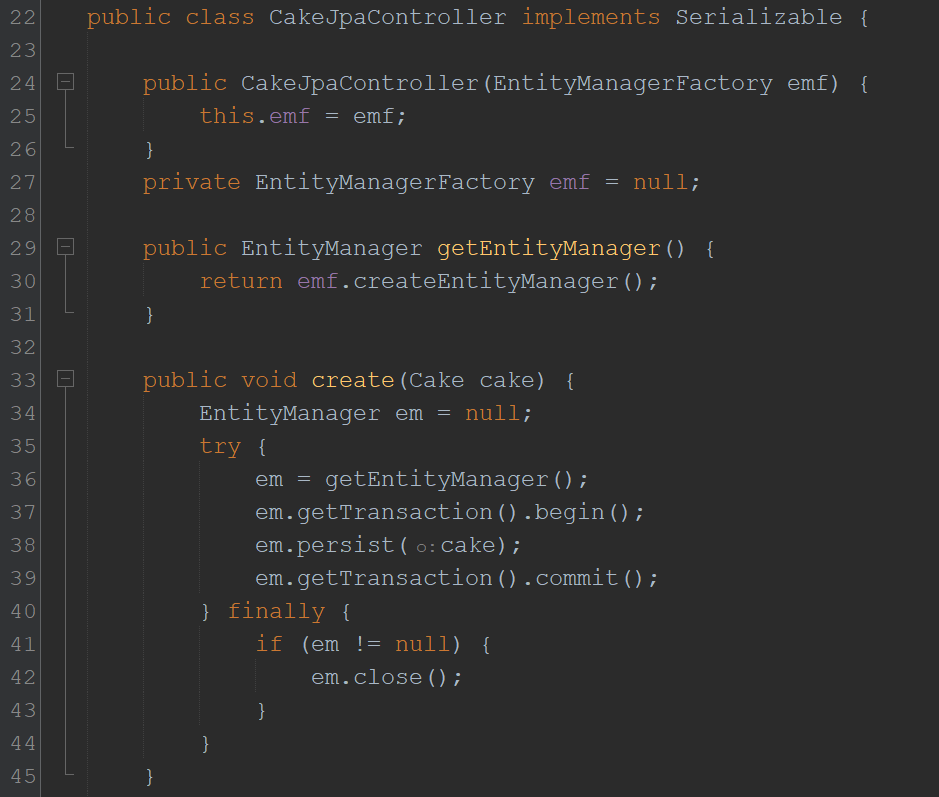


Graphical user interface, text

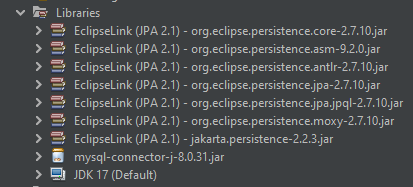
Description automatically generated







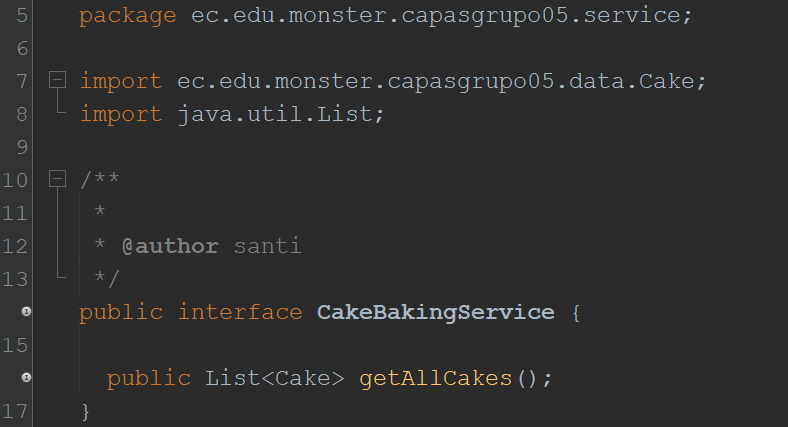
Para ejecutar el proyecto se debe agregar el conector de mysql al proyecto



Archivo de persistencia



Capa de servicio



Text

Description automatically generated

Capa de presentación

Text

Description automatically generated

Text

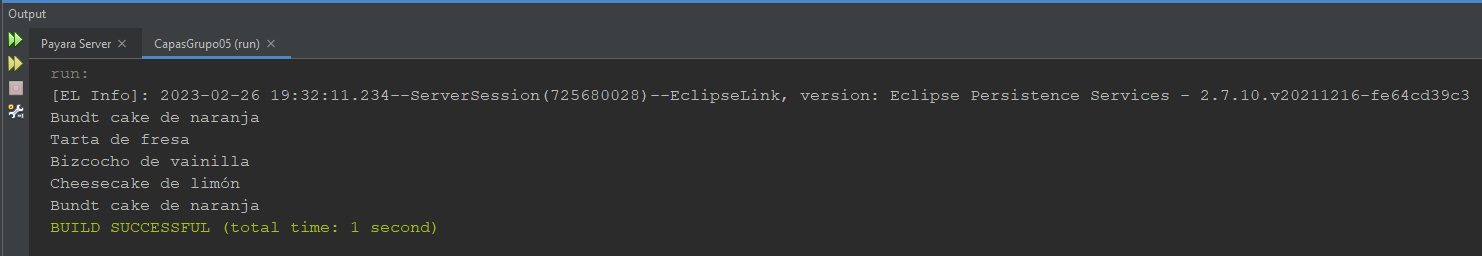
Description automatically generated

ESTRUCTURA FINAL DEL PROYECTO

Text

Description automatically generated

EJECUCION



**CREACION DEL PROYECTO MVC**

Considere la sala de UCI en el hospital que muestra la información de salud de los pacientes en las pantallas de los dispositivos que reciben información de los sensores conectados al paciente. Aquí, el trabajo de la pantalla es mostrar los datos que recibe del controlador, que a su vez se actualiza desde el modelo del sensor.

Text

Description automatically generated

ESTRUCCTURA INICIAL DEL PROYECTO

Graphical user interface, text, application

Description automatically generated

Capturas con las clases de cada paquete

Modelo

Vista

Controlador

ESTRUCTURA FINAL DEL PROYECTO

Graphical user interface, text, application, chat or text message

Description automatically generated

EJECUCION

Text

Description automatically generated

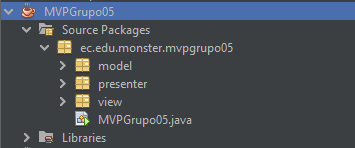
**CREACION DEL PROYECTO MVP**

Considere la aplicación de selección de archivos que permite seleccionar un archivo del almacenamiento. La lógica de selección de archivos está completamente separada de la implementación de la interfaz de usuario.

Text

Description automatically generated

ESTRUCTURA INICIAL DEL PROYECTO



CODIFICACION DEL PROYECTO

Poner capturas del código por cada paquete

ESTRUCTURA FINAL DEL PROYECTO

Text

Description automatically generated

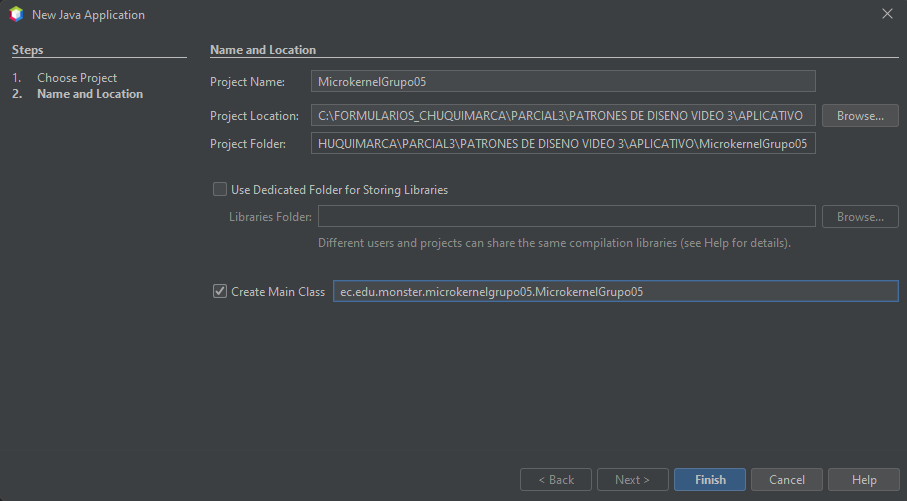
EJECUCION

Graphical user interface, text, application

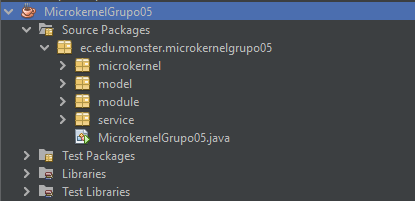
Description automatically generated

**CREACION DEL PROYECTO (MICROKERNEL)**

Supongamos que tenemos una aplicación que se encarga de procesar pedidos de diferentes tiendas en línea y queremos separar la lógica central de la aplicación de los diferentes módulos que manejan los pedidos para que estos puedan ser fácilmente intercambiables. En este caso, podríamos utilizar el patrón de diseño Microkernel para implementar una arquitectura modular y escalable. Con esta implementación, podemos agregar y quitar módulos de la aplicación de forma dinámica, lo que nos permite escalar y modificar la aplicación con facilidad. Por ejemplo, podríamos agregar un módulo para manejar los pedidos de otra tienda en línea simplemente implementando la interfaz Module y luego agregando el módulo a través del microkernel.



ESTRUCTURA INICIAL DEL PROYECTO



CODIFICACION DEL PROYECTO

MODELO

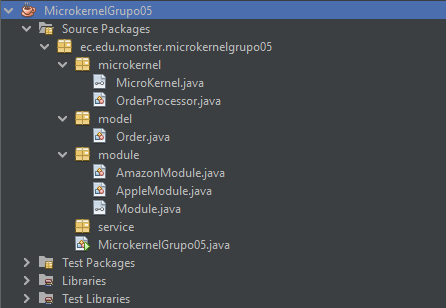
Poner capturas del código

MICROKERNEL

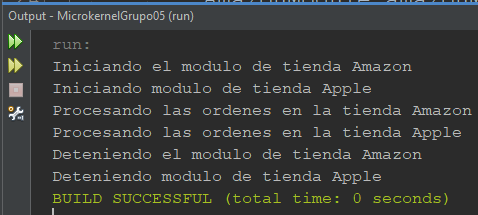
Poner capturas del código

MODULE Poner capturas del código

ESTRUCTURA FINAL DEL PROYECTO

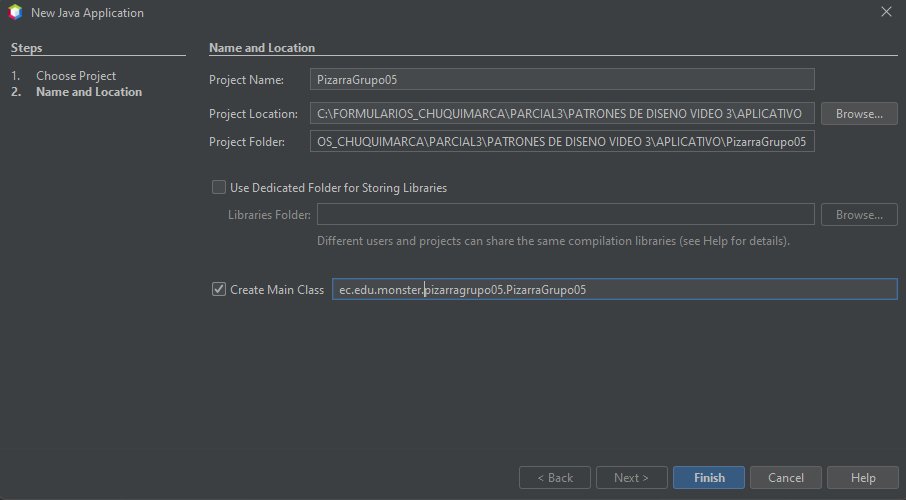


EJECUCION DEL PROYECTO

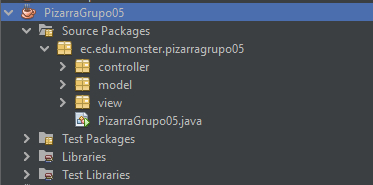


**CREACION DEL PROYECTO (PIZARRA)**

Un sistema de gestión de pedidos en línea. Este sistema permitirá a los clientes realizar pedidos de productos en línea, y a los empleados gestionar y completar los pedidos.

****

**ESTRUCTURA INICIAL DEL PROYECTO**

****

**CAPTURAS DEL CODIGO**

**ESTRUCTURA FINAL DEL PROYECTO**

**Graphical user interface, text, application, chat or text message

Description automatically generated**

**EJECUCION**

**Text

Description automatically generated**

**CREACION DEL PROYECTO (BROKER)**

En este ejemplo, creamos un objeto Broker y dos objetos Usuario. Agregamos los usuarios al broker y luego enviamos algunos mensajes utilizando el método enviarMensaje del broker. Finalmente, eliminamos los usuarios del broker.

Este ejemplo muestra cómo se puede utilizar el patrón Broker para implementar un sistema de mensajería en el que el broker actúa como intermediario entre los usuarios y se encarga de distribuir los mensajes entrantes a los usuarios correspondientes de manera eficiente.

**Text

Description automatically generated**

**ESTRUCUTRA INICIAL DEL PROYECTO**

**Graphical user interface, text

Description automatically generated**

**CAPTURAS DEL CODIGO**

**ESTRUCTURA FINAL DEL PROYECTO**

**Text

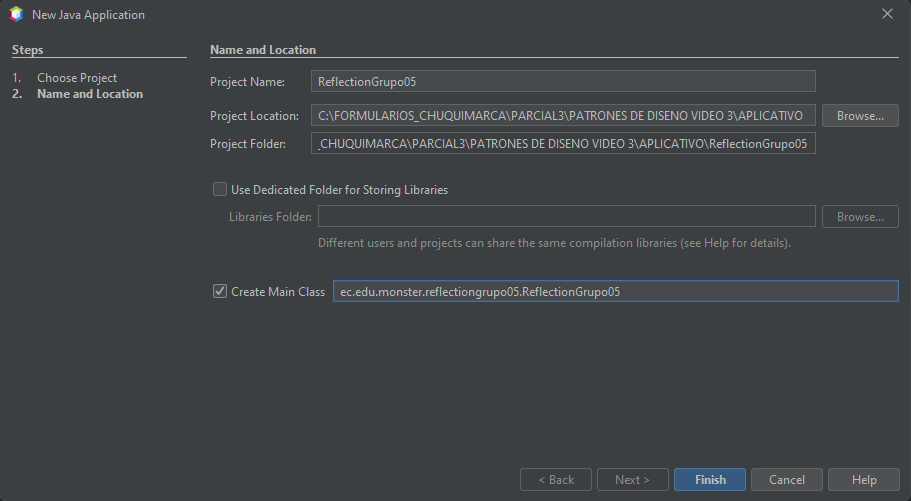
Description automatically generated**

**EJECUCION**

**Text

Description automatically generated**

**CREACION DEL PROYECTO(REFLECTION)**

En este ejemplo, utilizamos la reflexión para obtener referencias a los métodos de la clase Calculadora y llamarlos dinámicamente en tiempo de ejecución. Utilizamos el método estático Class.getMethod para obtener una referencia a un método específico de la clase Calculadora, y luego utilizamos el método invoke para llamar al método y pasarle los argumentos correspondientes.

**ESTRUCUTRA INICIAL DEL PROYECTO**

**Graphical user interface, text, application

Description automatically generated**

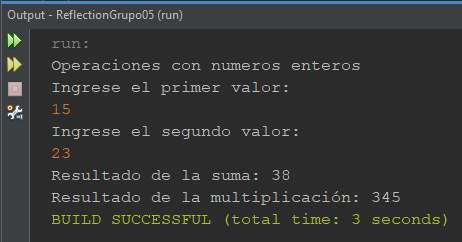
**CAPTURAS DEL CODIGO**

**ESTRUCTURA FINAL DEL PROYECTO**

**Graphical user interface, text, application

Description automatically generated**

**EJECUCION**

****