Ambito de decision (Objetos / Arquitectura / Persistencia / Otro)	Componente/s impactado/s	Decisión	Otras Alternativas	Justificación de la decisión
Lenguaje de Programación	Codigo del sistema	Utilizar JAVA	.net, C#, etc.	Teniamos mas conocimientos previos sobre JAVA y nos parecio el más adecuado para afrontar el TP.
Código	Main	Poner la menor cantidad de métodos posibles		Por ahora solo pusimos dos métodos, que son para pedir por pantalla: String o Int. Cada vez que necesitemos pedir algo por pantalla en otras clases, lo hacemos con estos. Además, llamamos a la clase singleton Sistema donde está el método para que empiece el programa.
Código	Sistema	Crear clase Sistema	No crear la clase	Decidimos crear la clase Sistema, con la funcion de ser el nucleo de nuestro sistema. En dicha clase, se instanciaran ambos gestores, ademas de enviarles mensajes a los mismos. En un futuro, sera la encargada de controlar todo el programa. No la agregamos al Diagrama De Clases porque no tiene más lógica que hacer empezar el programa.
Diagrama De Clases	Producto	Clase Producto y clase Item	No hacer la clase item y calcular directamente en Compra	Hicimos una clase producto, la cual tiene el producto junto con su proveedor, ya que un mismo producto puede ser fabricado por distintos proveedores. Por el otro lado, una clase item, que es la que se relaciona con la compra, y contará con un producto y la cantidad de dicho producto que se compro. Este último tendrá un método "Valor total" que calcula el Precio del Producto por la Cantidad de dicho Poducto que hay en la compra. Así, en la Compra se tendrá una lista de Items, y para saber el valor total de toda la compra habrá que hacer una sumatoria del valorTotal() de cada item.
Diagrama De Clases	Clase Compra	Dividimos a la clase compra en dos: Compra y Egreso	Considerra a la compra y al egreso como lo mismo	Consideramos que el egreso y la compra son dos cosas distintas. La compra es la que contiene todos los datos: items, presupuestos entre los que puede elegir, usuarios revisores suscriptos a la bandeja de mensajes de esa compra, documentos, etc. Además, tiene los métodos tales como el encargado de elegir el presupuesto, el valor total (que será el del presupuesto elegido), elegir la estrategia en base a la cual se eligirá el presupuesto, etc. Por último, tenemos un método efectuarCompra() que crea un nuevo egreso.
Diagrama De Clases	Clase Egreso	Dividimos a la clase compra en dos: Compra y Egreso	Considerra a la compra y al egreso como lo mismo	A nuestro criterio, el Egreso será la compra efectuada, por eso es que se crea cuando se ejecuta el metodo efectuarCompra() en la clase Compra, y se le pasará la fecha actual. El egreso contendrá, por lo tanto, una compra y su fecha de pago. El Gestor de Egresos es el que se encargará de hacer más cosas que el Egreso en sí.

Diagrama De Clases	Presupuesto	Agregamos la clase Presupuesto		La compra contiene un presupuesto. El presupuesto contiene los mismos items que la compra, los documentos comerciales que contiene una compra, y el valor total del presupuesto. Asumimos que una compra contendrá dentro de su lista de Presupuestos, únicamente Presupuestos que tengan los mismos items que ella. Si no no tendría sentido que lo considere.
Codigo	Validador	Singleton		Cuenta con una variable que es la de presupuestos máximos. Esta es setteada. Es la cantidad de presupuestos máximos que puede tener una compra. También tiene los tres métodos que validan lo pedido en el enunciado de la Entrega 2. La compra tendrá un método validar() que utilizará dicho validador. Lo que hace es validar y guardar el resultado de la validación en la bandeja de mensajes del Usuario suscripto.
Codigo	Compra	Lista de usuarios revisores		Cada compra tendrá una lista de Usuarios revisores. Cuenta con un metodo para hacer que a cada uno de ellos les llegue la validación. Agregamos la variable booleana requierePresupuesto ya que si no lo requiere, hay varias cosas que no se deben validar. El metodo validar llama a los validadores dentro de la clase Validador.
Diagrama De Clases	Criterio	Agregamos la clase Criterio, utilizando el patrón Strategy, para los Criterios en base al cual se eligirá el Presupuesto	Elegir un presupuesto dentro de la clase Compra, usando IFs	Por ahora solo tenemos un criterio que nos lo dice el enunciado y ese es el de menor valor. Sin embargo, asumimos que en el futuro habrán más. El método seleccionarPresupuesto() dentro de Compra lo que hace es llamar al método elegriPresupuesto dentro de su Criterio elegido.
Código	Verificar password	Verifica que se cumplan todas las restricciones mencionadas en la Entrega 1 sobre las contrsaeñas.		Verifica que no se encuentre entre las 10.000 peores, que tenga entre 8 y 64, que la seguridad sea mayor o igual a 80 (esto implica que tenga minusculas, una mayuscula, un numero y un simbolo (al menos)), y que no sea igual al nombre de usuario, si no cumple alguna, pide otra. La verificación de contraseña se realiza en el Gestor de Password.
Archivo de texto	10.000 peores contraseñas	10.000 peores contraseñas encontradas en Wikipedia		Wikipedia fue el único lugar donde encontramos 10.000 peores contraseñas, el resto solían ser mucho menos, y por más que estén en inglés usamos este archivo. En un futuro podremos traducirlas.

Código	Hashear contraseña	Método SHA-384		Fue el algoritmo que nos resultó más fácil de entender.
Diagrama De Clases	Categorizador	Agregamos clase Categorizador		Esta clase se encarga de categorizar el tipo de empresa (Micro, Pequeña, Mediana Tramo 1 o Mediana Tramo 2) a partir de el promedio de ventas anuales, el personal asignado, y el sector al que pertenece (datos ingresados al darse de alta la empresa). El categorizador se ejecuta apenas se da de alta la empresa.
Diagrama De Clases	Entidad	Composite	Herencia	Consideramos que aplicar el patron de diseño composite, era mas adecuado para resolver el problema que simplemente usando herencia. Ya que el enunciado nos aclara que las entidades juridicas estan compuestas por las entidades bases, y estas a su vez, son entidades.
Código	Gestor de Usuarios/Password	Crear clases Singleton	No abstraerlo en clases	Preferimos crear una clase Gestor de Usuarios y Gestor de Password. La clase Gestor De Usuarios, se encarga de controlar todo lo relacionado con los usuarios, mientras que la clase Gestor De Password, se encarga de validar y crear las nuevas Passwords. Ninguna de las dos clases las agregamos al Diagrama de Clases porque no pertenecen a la lógica de negocio.
Codigo	Gestor de Egresos	Singleton		Contiene una lista de Compras y una lista de Presupuestos. Se encarga de registrarlos a partir de los métodos en la clase Egresos. También tiene un método que busca compra por ID. Esto es para que el Usuario pueda subscribirse como validador/revisor de dicha compra.
Codigo	Interfaz	Abstraimos de la parte de negocio, lo relacionado con la lógica de interfaz, creando dos clases nuevas: Interfaz Usuario e Interfaz Password		La decisión se debe a que no es correcto mezclar la capa de negocio con la capa de interfaz.