| **Ambito de decision (Objetos / Arquitectura / Persistencia / Otro)** | **Componente/s impactado/s** | **Decisión** |
|---|---|---|
| Lenguaje de Programación | Codigo del sistema | Utilizar JAVA |
| Archivo de texto | 10.000 peores contraseñas | 10.000 peores contraseñas encontradas en Wikipedia |
| Diagrama De Clases | Entidad | Composite |
| Diagrama De Clases | Producto | Clase Producto y clase Item |
| Código | Sistema | Crear clase Sistema |
| Código | Gestor de Usuarios/Password | Crear clases Singleton |
| Código | Main | Poner la menor cantidad de métodos posibles |

| | | |
|---|---|---|
| Código | Verificar password | 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. |
| Código | Hashear contraseña | Método SHA-384 |

| Otras Alternativas | Justificación de la decisión |
|---|---|
| .net, C#, etc. | Teniamos mas conocimientos previos sobre JAVA y nos parecio el más adecuado para afrontar el TP. |
| | 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. |
| 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. |
| 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, y una clase item, que es la que se relaciona con la compra, y contará con la cantidad de ese item que se compro. Este último tendrá un método "Valor total" que calcula el Precio del item por la Cantidad de dicho item que hay en la compra. |
| 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 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. |
| | 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. |

| | |
|---|---|
| | |
| | Fue el algoritmo que nos resultó más fácil de entender. |