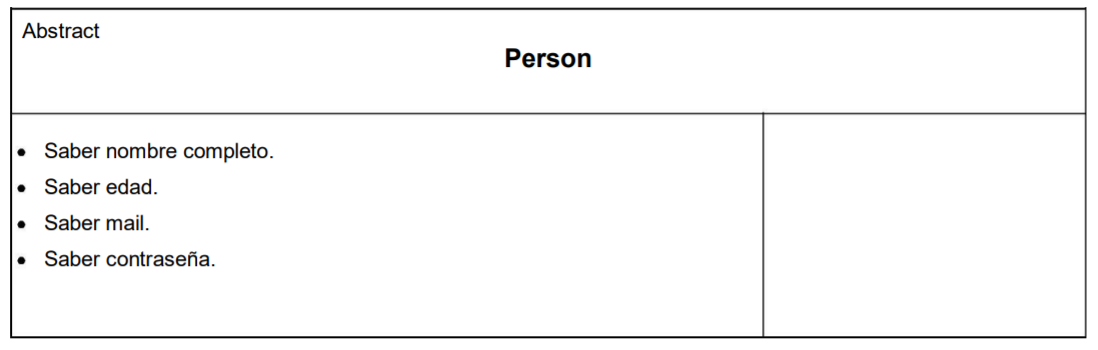
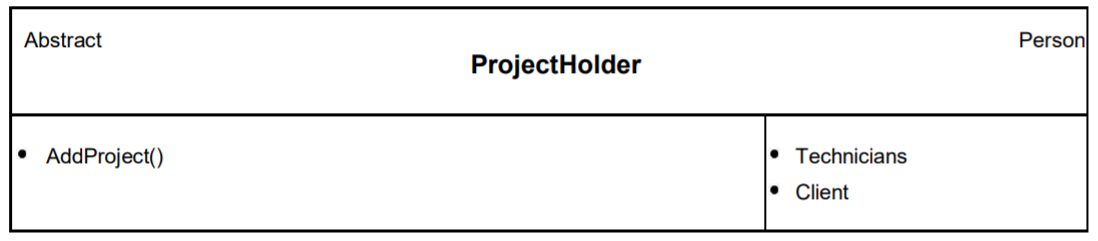
Justificación de clases

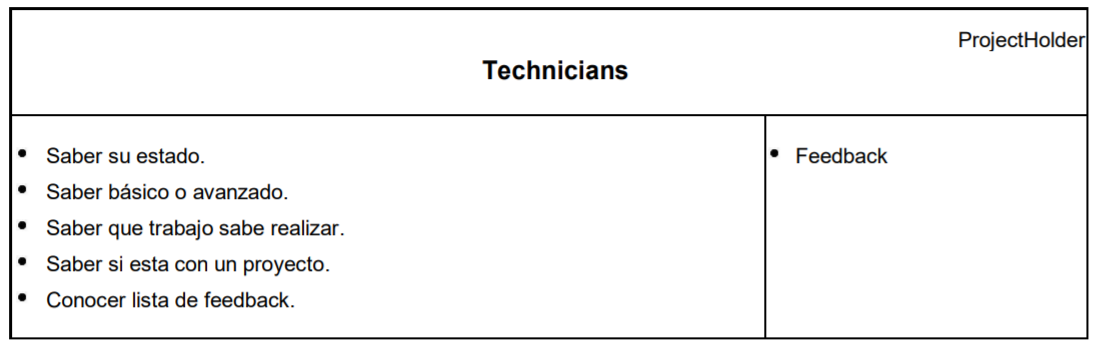
Model



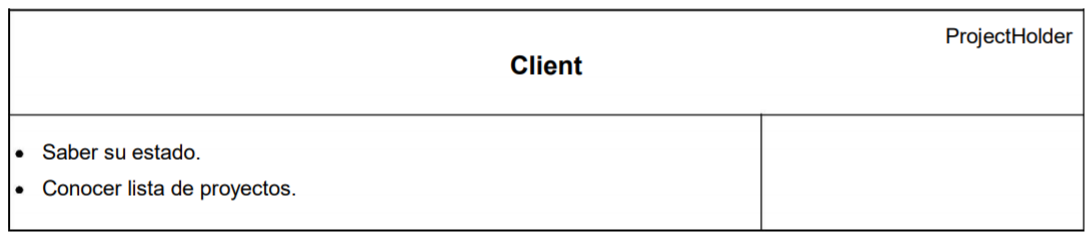
Creamos la interfaz Person como base ya que existen tres clases que comparten atributos más generales (relación degeneralización-especialización) y a su vez esto favorece la reutilización de código. Esta interfaz también nos sirve en caso de que se quiera crear un tipo de usuario nuevo en el futuro, esta abierta a la extensión y cerrada a la modificación (open/closed principle).



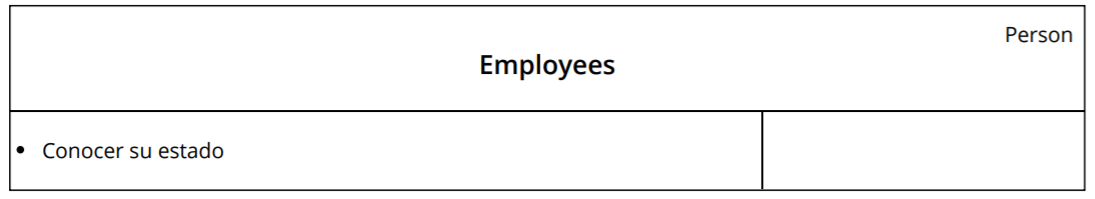
La clase abstracta ProjectHolder hereda de Person, esta fue creada porque favorece la reutilización de código en Technicians y Client las cuales comparten el método AddProject. El método AddProject no está en Person ya que no necesariamente una persona tiene que poder agregar un proyecto (principio de sustitución de Liskov).



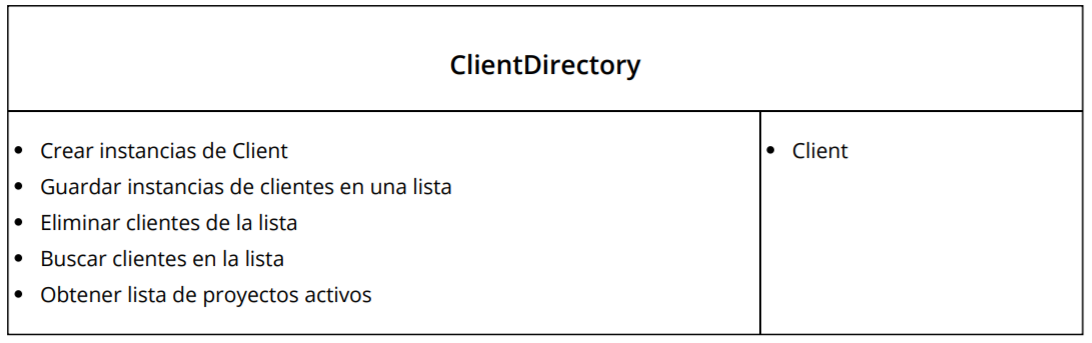
Technicians hereda de ProjectHolder, tiene los atributos de Person e implementa método AddProjet de ProjectHolder; a su vez tiene atributos más específicos de los técnicos.



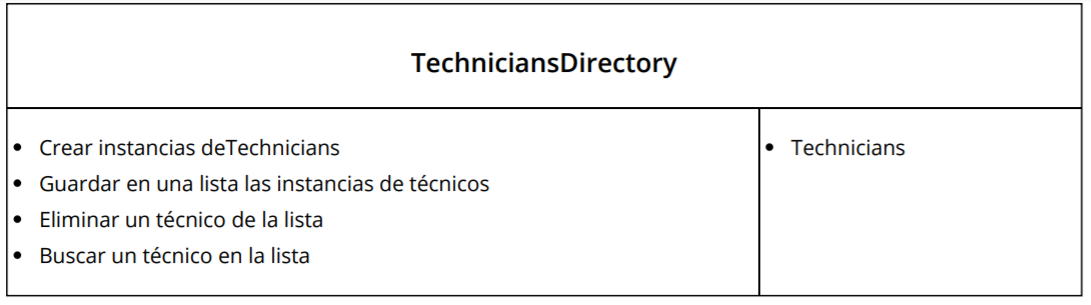
Client hereda de ProjectHolder, conoce su estado y a su vez implementa el método AddProject.



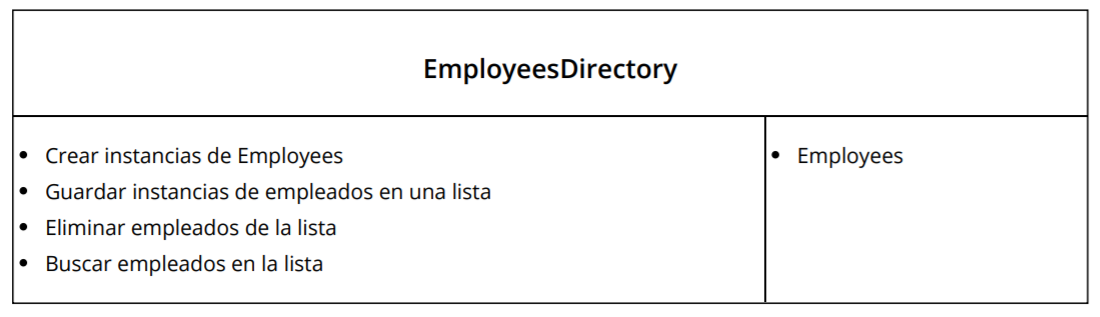
Employess hereda de Person y conoce su estado.



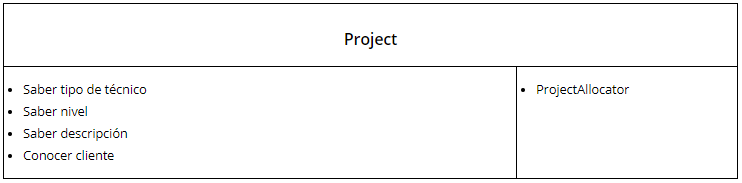
Esta clase es la que crea las instancias de clientes ya que es la que tiene los datos necesarios para ello por esto cumple con el patrón Expert y a su vez cumple con el patrón Creator ya que instancia y almacena la información de clientes(buscar).



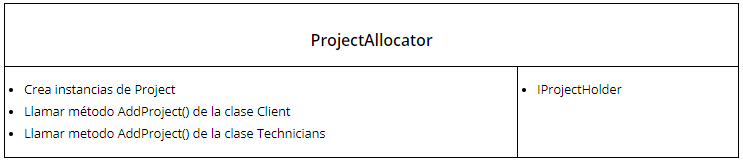
Esta clase es la que crea las instancias de tecnicos ya que es la que tiene los datos necesarios para ello por esto cumple con el patrón Expert y a su vez cumple con el patrón Creator ya que instancia y almacena la información de tecnicos(buscar).



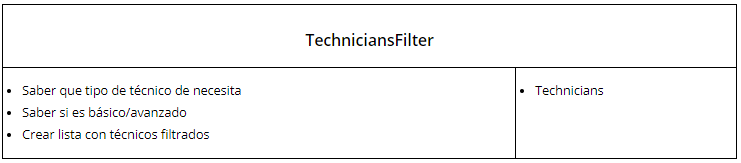
Esta clase es la que crea las instancias de empleados ya que es la que tiene los datos necesario s para ello por esto cumple con el patrón Expert y a su vez cumple con el patrón Creator ya que instancia y almacena la información de empleados(buscar).



La responsabilidad Project es la de conocer su estado.

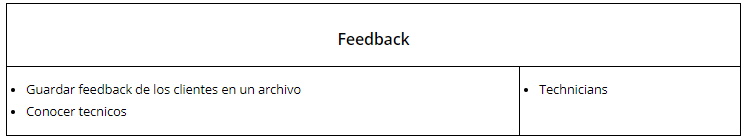


Esta clase es la encargada de crear instancias de proyectos ya que es la que tiene los datos necesarios ( Expert). A su vez llama al método AddProject de Client para agregar el proyecto a la lista; una vez que un técnico acepta un proyecto lo que hace ProjectAllocator es agrégalo a Technicians con el método AddProject.

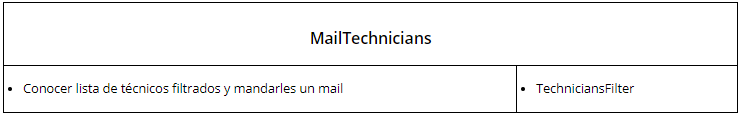


Esta clase es la encargada de filtrar a los tecnicos ya que sabe los datos necesarios (Expert)

A su vez crea una lista con los tecnicos filtrados.

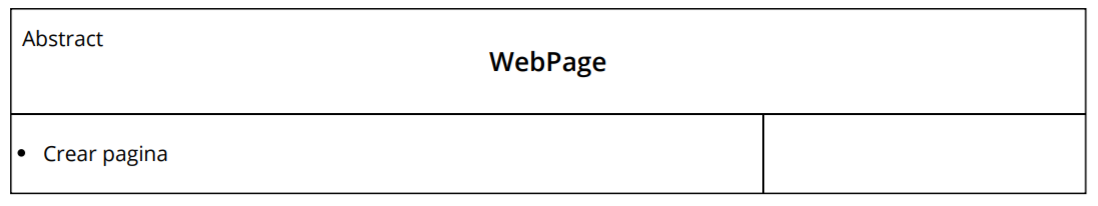


La responsabilidad de esta clase es guardar el feedback de los clientes en un archivo y asignarlo a cada técnico. Esta clase cumple con el Expert ya que tiene todos los datos necesarios para asignarle el feedback a los técnicos.

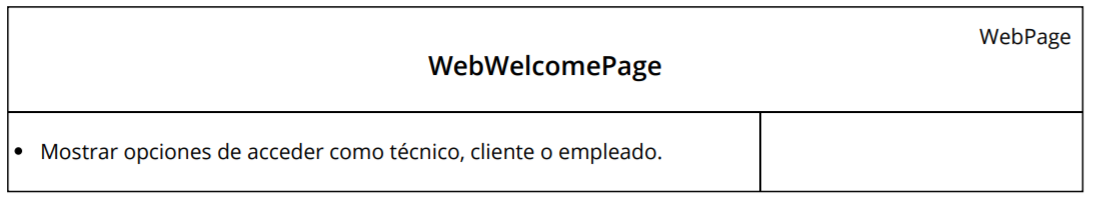


Esta clase colabora con TechniciansFilter por lo que conoce la lista de tecnicos filtrados y les manda un mail a esos tecnicos(Expert)

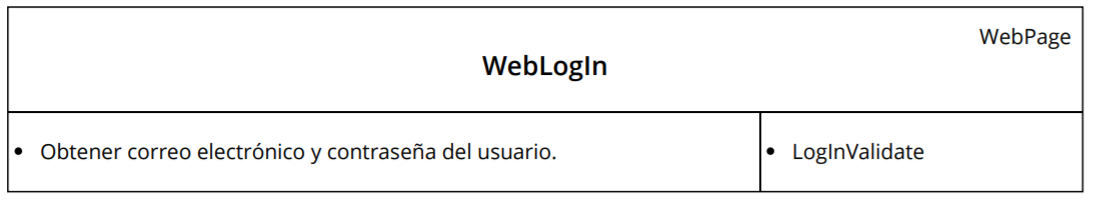
View



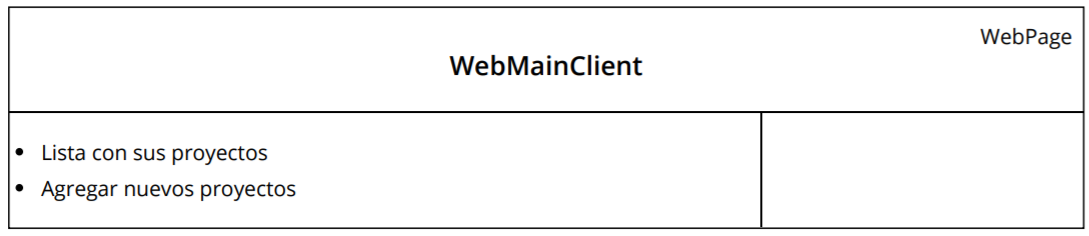
Creamos la clase abstracta WebPage como base para la creación de la página (Herencia), esto a su vez favorece la reutilización de código. Cumple con el Open/Closed Principle, ya que esta abierta a la extensión y cerrada a la modificación.



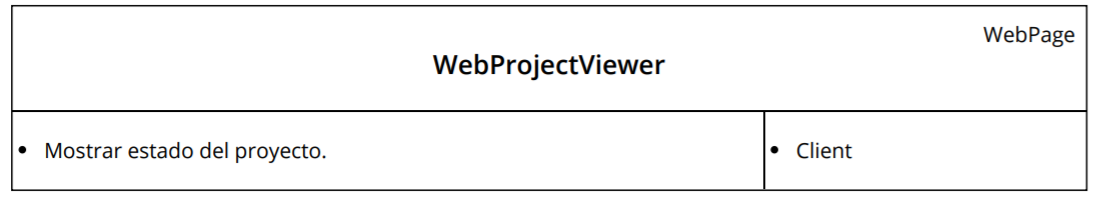
Esta clase hereda de WebPagela cual crea la pagina y WebWelcomePage permite elegir si entrar como tecnicio, cliente o empleado



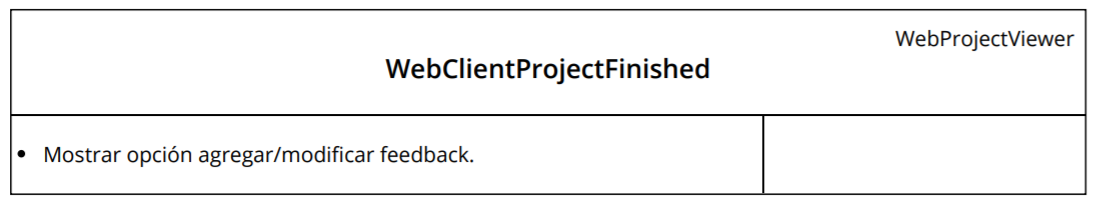
Esta clase la cual usa el motodo de LogInValidate(Delegacion) para verificar que los datos corresponden con los datos ingresados para el Login.



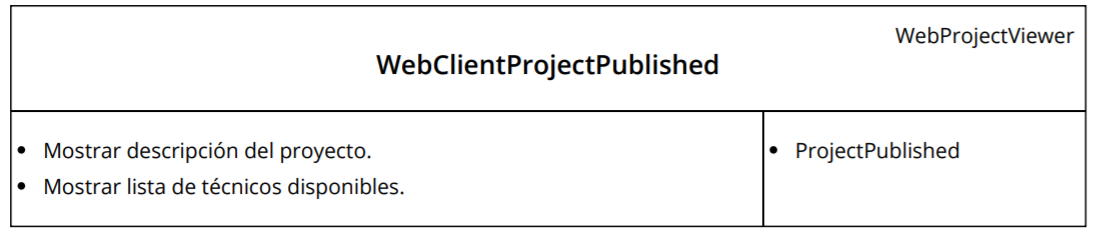
Esta clase que hereda de WebPage la cual ve el Cliente al entrar, en la cual puede visualizar una lista con sus proyectos y tiene la opcion de agregar nuevos.



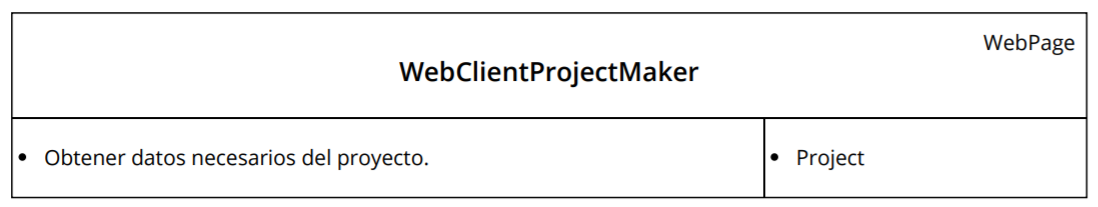
Esta clase colabora con Client por lo que sabe el estado del proyecto y lo muestra (Expert).



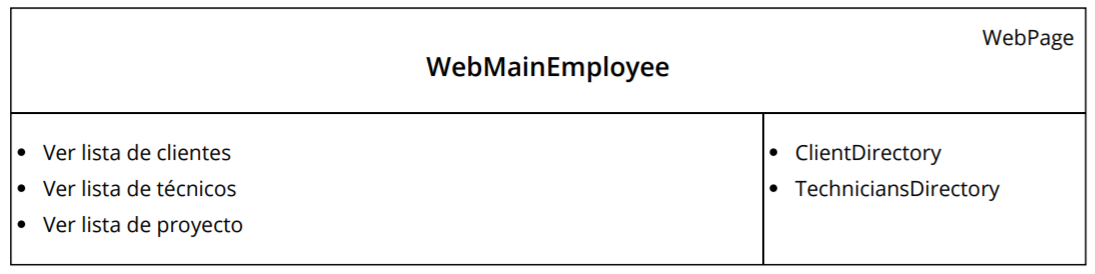
Esta clase hereda de WebProjectViewer y al mostrar un proyecto terminado da la opción de agregar o modificar feedback(Expert).



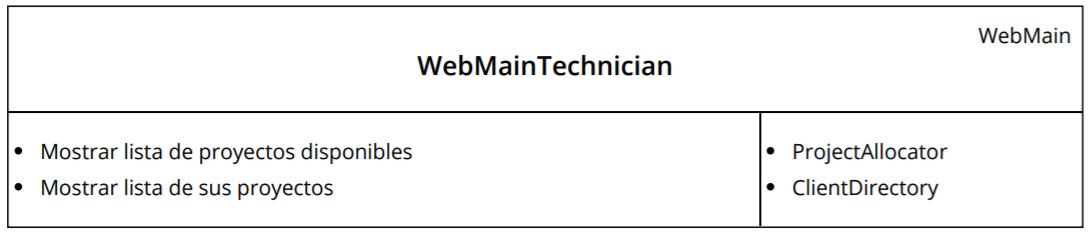
Esta clase hereda de WebProjectViewer y colabora con ProjectPublished ya que al entrar a un proyecto ya publicado muestra la descripción y una lista con los técnicos disponibles para el proyecto.



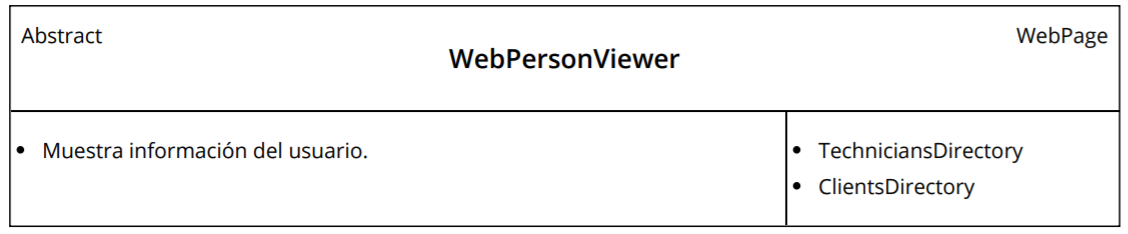
Esta clase colabora con Project por lo que conoce el estado y muestra los datos (Expert).



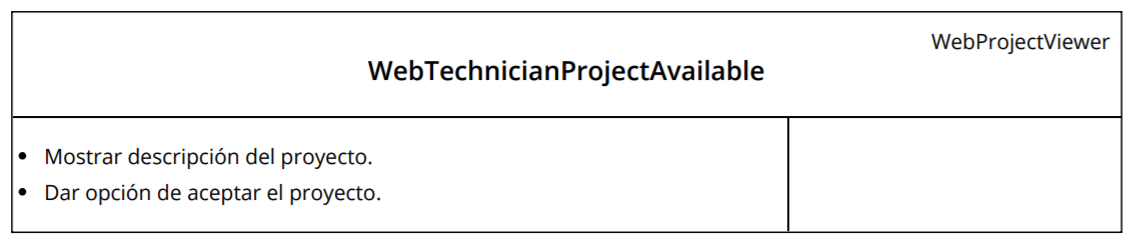
Esta clase colabora con ClientDirectory y TechnicianDirectory para mostrar una lista con los clientes, una con los técnicos y otra con los proyectos.



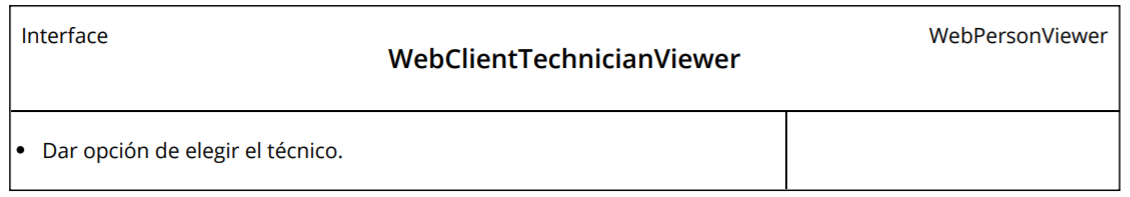
Esta clase colabora con ProjectAllocator y ClientDirectory para mostrar la lista de proyectos disponibles y mostrar la lista de proyectos en la que participa o ha participado.



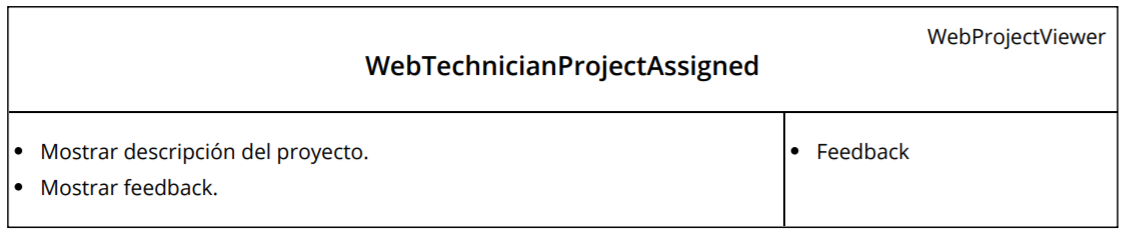
Creamos la clase abstracta WebPersonViewer para que todas las páginas que muestran información de un usuario puedan heredar de ella, cumpliendo con el principio de responsabilidad única, la única responsabilidad por la que podría cambiar es si cambiara la forma en la que se muestra la información de un usuario. Esta abierta a la extensión (agregar nuevas clases de vista de usuarios) y cerrada a la modificación y cumple con el principio de sustitución de Liskov (todas las páginas de las que hereda pueden cumplir su lugar).



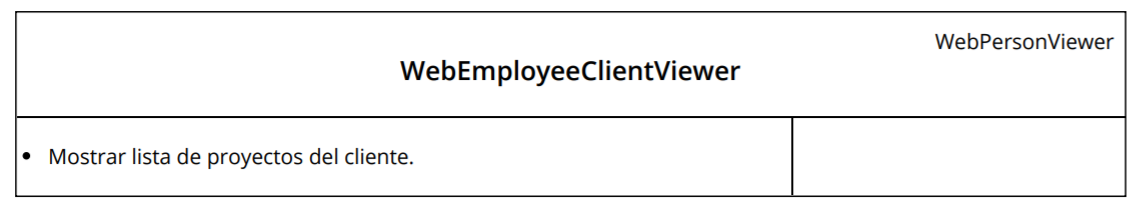
Esta clase hereda de WebProjectViewer y además da la opción de aceptar un proyecto, siendo esta su única razón de cambio (principio de responsabilidad única).



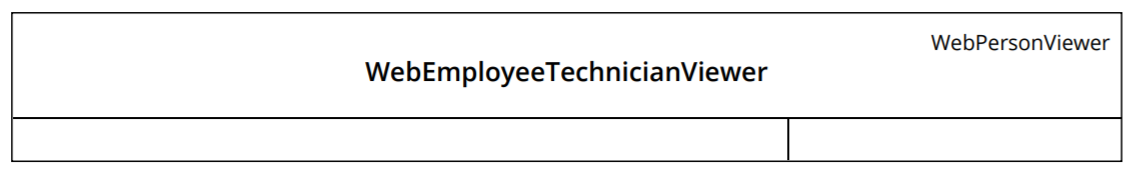
Esta clase hereda de de WebPersonViewer, da la opción de elegir al técnico siendo esta su única razón de cambio (principio de responsabilidad única).



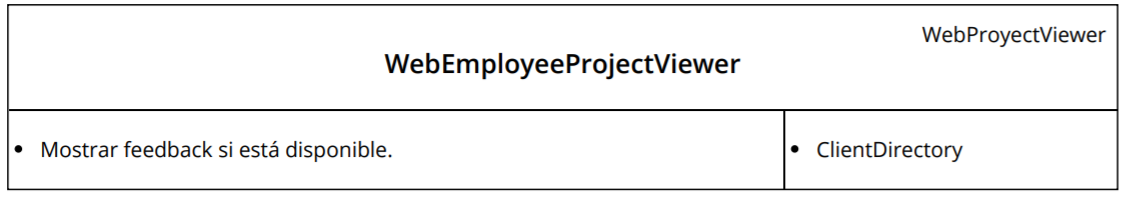
Esta clase hereda de WebProjectViewer y colabora con Feedback para mostrar el feedback del proyecto. Cumple con el principio de sustitución de Liskov y con el principio de responsabilidad única.



Esta clase colabora con WebPersonViewer y muestra la lista de proyectos del cliente.

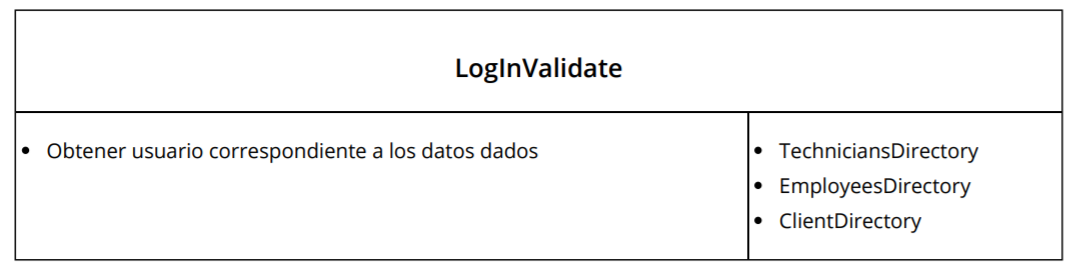


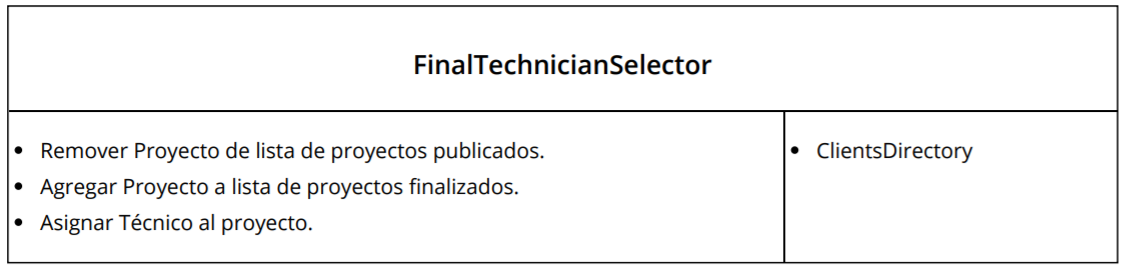
Esta clase hereda de WebPersonViewer. La creamos para cumplir con el principio de inversión de dependencias.



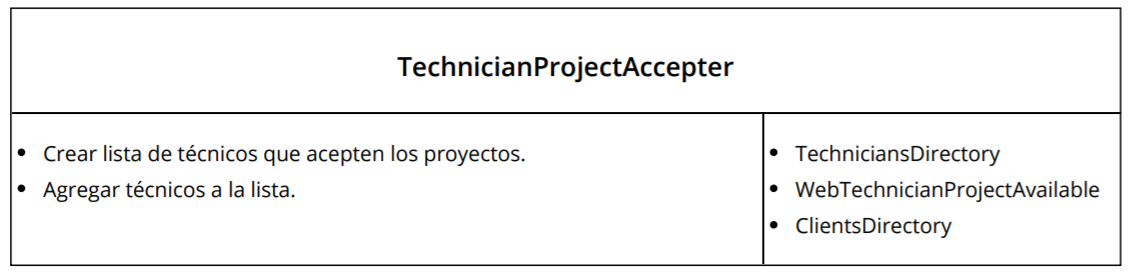
Esta clase hereda de WebProjectViewer y colabora con ClientDirectory para mostrar el feedback (si está disponible). Cumple con el patrón Expert, ya que es la clase que tiene toda la información necesaria (feedback) para mostrarle al empleado.

Control

Esta clase colabora con TechniciansDirectory, EmpleyeesDirectory y ClientDirectory por lo que sabe los datos de los mismos y verifica si los datos ingresados son los conrrespondientes a la instancia(Expert).



Esta clase colabora con ClientsDirectory para asignar un Técnico a un proyecto y moverlo de la lista de proyectos publicados a los asignados (patrón Controller).



Esta clase crea la lista de técnicos que aceptan un proyecto y los agrega a ésta, por lo que es la experta en la responsabilidad y cumple con el patrón Creador. También cumple con el patrón de responsabilidad única, el único motivo por el que cambiaria es si en un futuro cambiamos la forma en la que los técnicos aceptan los proyectos.