

Está Vd. en: INICIO - Menú Inicial

Usuario: cpompa

Ayuda













Práctica de la asignatura ISG3

Autores: Macarena Rodríguez Rodríguez Carlos Pompa Domínguez





01	¿Qué es SVL?	3
02	Especificaciones Técnicas	7
03	Herramientas Software Utilizadas	7
04	Subsistemas Identificados	12
05	Prototipo Web	18
06	Casos de Uso implementados	12
	1. Registro de Voluntarios	12
	Gestión de Alertas	12
07	Modelo Conceptual	20
80	Diseño de Clases	20
09	DAO	20







Servicio de Voluntariado Local

¿Qué es SVL?

SVL (Servicio de Voluntariado Local) es una propuesta de portal web cuyo principal objetivo es facilitar el acceso a los <u>servicios de tipo asistencial</u> de aquellas personas que lo necesitan.

La idea principal es que dicho servicio sea de <u>ámbito local</u>, es decir, que dentro de una misma localidad pueda confluir la oferta y la demanda de dichos servicios.

Se trata básicamente de que todas las personas que quieran dedicar su tiempo y sus conocimientos a servir a otras personas de forma voluntaria, se registren en la web como voluntario, para que aquellos demandantes registrados a su vez, puedan encontrarlos.

Es decir, el fin último de la web es confluir oferta y demanda de servicios de voluntariado.





Especificaciones Técnicas



Especificaciones Técnicas

Las especificaciones técnicas del desarrollo del aplicativo son las siguientes:

- ❖ <u>Servidor de Aplicaciones:</u> **Tomcat 5.5**.
- ❖ Versión JAVA: JDK 1.6.0
- ❖ Base de datos: MySQL Server 5.5
- ❖ Interfaz de Usuario: Páginas JSP y HTML con Javascript
- ❖ Patrón del Controlador: Front-Controller
- ❖ Arquitectura en 3 capas MVC.





Herramientas Software Utilizadas

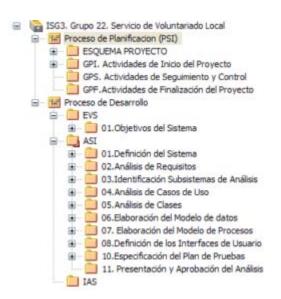


Servicio de Voluntariado Local

Herramientas Software Utilizadas

Las herramientas Software utilizadas en el desarrollo del aplicativo son las siguientes:

- ❖ IDE: Eclipse Helios Services Release 2
- ❖ Enterprise Architect: Para el análisis y diseño del proyecto. Esta herramienta permite generar toda la documentación asociada a estas 2 fases.



- ❖ Macromedia Dreamweaver: Para el desarrollo del Prototipo Web.
- * Adobe Photoshop: Para el diseño de la web y el tratamiento de imágenes asociadas.





Subsistemas Identificados



Servicio de Voluntariado Local

PROTOTIPO WEB

Los principales subsistemas identificados en el aplicativo son los siguientes:

Voluntarios

Este subsistema permite la gestión de la información de los Voluntarios. Además se incluye un formulario de alta para que las personas que quieran ofrecer sus servicios puedan hacerlo de una forma rápida y sencilla.

Demandantes

Este subsistema permite la gestión de la información de los Demandantes. Se incluye un formulario de alta para que las personas que requieren de estos servicios puedan solicitarlo de una forma rápida y sencilla.

Gestión de Alertas

Este subsistema permite la gestión de las alertas sobre los diferentes servicios que se ofrecen. De este modo, se podrá avisar a los demandantes (vía sms o correo electrónico) de los servicios disponibles y que se adecúan a sus necesidades.

Administración

Este subsistema permite la gestión interna del aplicativo. Incluye todo lo referente a la gestión de perfiles y usuarios del sistema, así como la parametrización de las tablas maestras.







PROTOTIPO WEB

El prototipo web es una versión "cartón piedra" y reducida de la aplicación hecha con **HTML** y **Javascript** en base a las especificaciones del aplicativo.





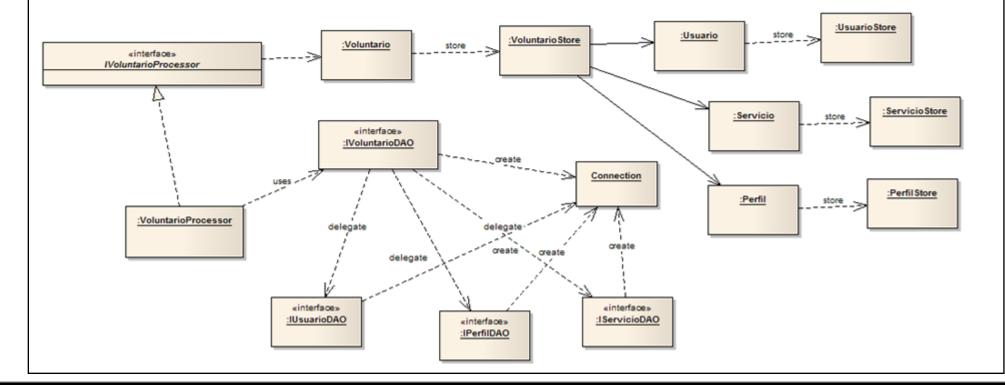


Casos de Uso implementados



Caso de Uso 1 - REGISTRO DE VOLUNTARIOS

+				
	Business Object	Transfer Objects	Data Access Objects	Data Source(s)
	IVoluntarioProcessor/ Ivoluntario registrarVoluntario(Usuario): void	Voluntario	IVoluntarioDAO.registrarVoluntario(Usuario usuario) IUsuariioDAO.insertarUsuario(Usuario usuario)	Voluntario Usuario Perfil
			IPerfilDAO.insertarPerfil(int usua_id) IServicioDAO.insertarServicio(Servicio servicio);	Servicio



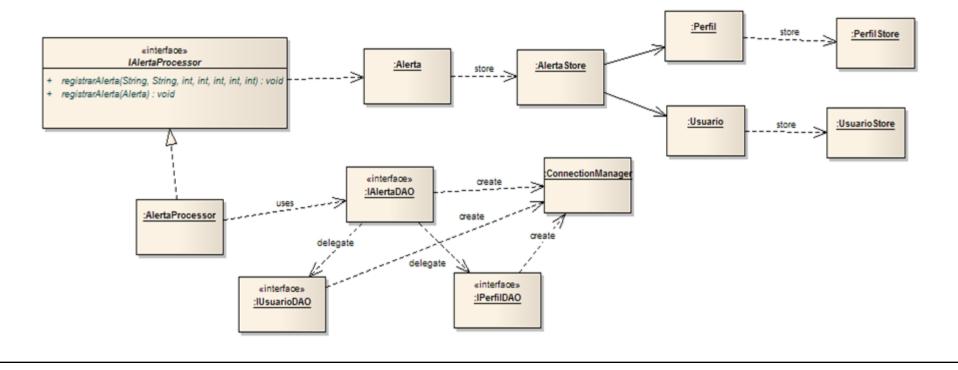


Caso de Uso 1 - REGISTRO DE VOLUNTARIOS			
UC- <u>0001</u>	Registro de Voluntarios		
Versión	1.0 (30	0-08-2011)	
Autores	Carlos F	Pompa Domínguez (Grupo 22)	
Fuentes			
Objetivos asociados			
Requisitos asociados Registro de Voluntarios		o de Voluntarios	
Descripción	Los usuarios anónimos se registran como voluntarios.		
	Para ello tienen que dar de alta sus datos personales y después los servicios en los que se		
	van a dar de alta como voluntario.		
Precondición	El usuario no está registrado previamente.		
Secuencia normal	Paso	Acción	
	P1	El usuario solicita darse de alta	
	P2	El sistema solicita los campos obligatorios para el registro	
	P3	El usuario rellena los datos y los envía	
	P4	El sistema registra los datos e informa de que los datos han sido registrados.	
	P5	El sistema muestra la lista de servicios opcionales a registrar.	
	P6	El usuario informa los servicios en los que se quiere dar de alta.	
	P7	El sistema registra los datos e informa de que los datos han sido registrados.	
Post-condición	El usuario es registrado en el sistema como voluntario y han sido de alta los servicios.		
Excepciones Paso Acción		· ·	
	P1	Si el usuario no rellena los servicios sus datos si quedan registrados. Pudiendo editar	
		luego los servicios.	



Caso de Uso 2 - GESTIÓN DE ALERTAS

Burdinas Object	Town of an Object	Data Assaus Objects	0-4-0
Business Object IALertaProcessor/ IAlerta registrarAlerta(String aler_nombrecontacto,String aler_emailcontacto,int prov_id, int muni_id, int serv_id, int tibu_id, int tise_id);		Data Access Objects IVoluntarioDAO.addAlerta(String usua_usuario, String usua_email); IUsuariioDAO.insertarUsuario(Usuario usuario) IPerfiIDAO.insertarPerfil(int usua_id)	Data Source(s) Alerta Usuario Perfil





Caso de Uso 2 – GESTIÓN DE ALERTAS			
UC- <u>0002</u>	Gestión de Alertas		
Versión	1.0 (30-08-2011)		
Autores	Macarena Rodríguez Rodriguez (Grupo 22)		
Fuentes			
Objetivos asociados			
Requisitos asociados Registro de Voluntarios y Servicios		Voluntarios y Servicios	
Descripción	Los usuarios anónimos solicitan dar de alta una alerta en el sistema.		
	Para ello tie	ra ello tienen que dar de alta un usuario y un email. Al dar de alta la alerta se registra con	
	los datos m	datos mínimos y se da de alta con el perfil no registrado. No se dan de alta servicios.	
Precondición	El usuario no está registrado previamente.		
Secuencia normal	Paso	Acción	
	P1	El usuario solicita dar de alta una alerta	
	P2	El sistema solicita los campos obligatorios para la creación de una alerta.	
	P3	El usuario rellena los datos y los envía	
	P4	El sistema registra los datos e informa de que los datos han sido registrados.	
	P5	El sistema registra como clave el nombre del usuario.	
	P6	El sistema registra los datos e informa de que los datos han sido registrados.	
Post-condición	La alerta es	alerta es dada de alta en el sistema.	
Mejoras	Paso	Acción	
	P1	Mandar un correo al usuario con la alerta dada de alta.	





Modelo Conceptual



Modelo Conceptual class 051.Modelo Conceptual del Dominio Usuario Servicio de Voluntariado Provincia muni_muniid: int prov_id: String prov provid: int Grupo 22 prov_nombre: String usua_apellidos: String ISG3 2010-2011 usua_clave: String usua_codpostal: Integer vive -Municipio usua_direction: String usua_email: String muni_id: String usua_esadmin: boolean muni_nombre: String usua_esdemandante: boolean prov_provid: String TipoBusqueda usua_esvoluntario: boolean usua_falta: Date tibu_id: String usua fbaja: Date titu_descrico: String usua_genero: int usua_id: String usua_nif: String usua_nombre: String usua_numero: String enla_descrico: String usua telefono: String + regi finicio: Date enla_descrila: String enla_id: String usua_usuario: String + regi_horafin: String + regi_horainicio: String Perfil enla_url: String regi_id: String perf descrico: String regi_totalhoras: int perf_descrila: String posee + usua_usuaid: String perf_id: String volu_volucod: String propone 0..-Servicio Voluntario UsuarioAnonimo serv_descrico: String usua_usuaid: String serv_descrila: String propone od: String genera inscribe aler descripcion: String serv_id: String volu falta: Date aler_emailcontacto: String tise_tiseid: String volu_fbaja: Date genera aler_id: String volu_id: String aler_nombrecontacto: String muni_muniid: int inscribe prov_provid: int serv_servid: int tibu_tibuid: int Demandante tise_tiseid: int demanda dema_cod: String dema_falta: Date dema_fbaja: Date Tipo Servicio dema_id: String usua_usuaid: String tise_descrico: String tise_descrila: String tise_id: String





Diseño de Clases



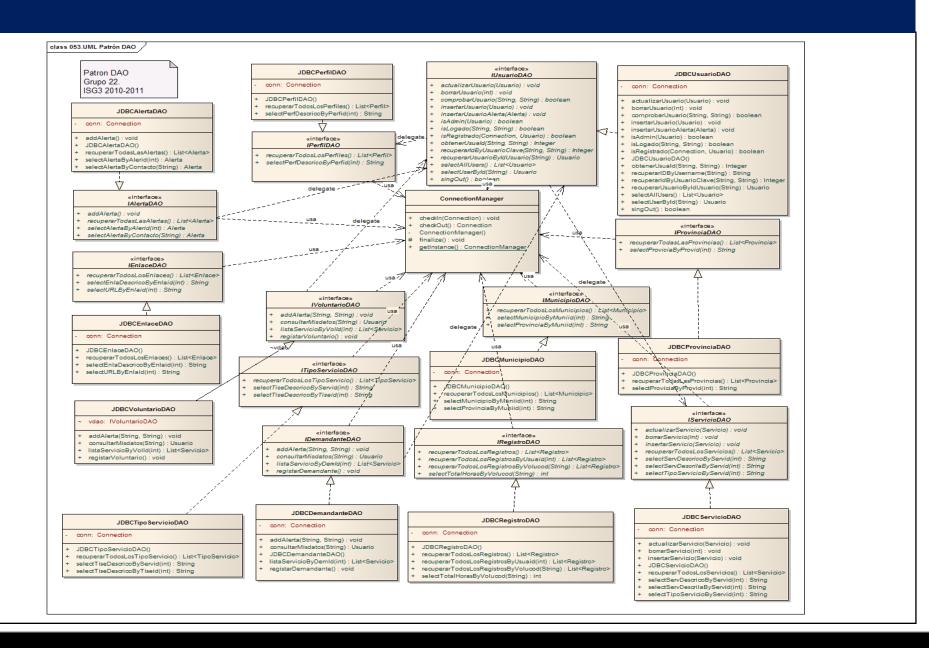
Diseño de Clases class 052.UML Diseño de Clases de Dominio IAIertaProcessor Alerta Store AlertaProcessor Local dao: JDBCAlertaDAO = new JDBCAlertaDAO(aler descripcion: String regi finicio: Date Grupo 22 aler_emailcontacto: String dao: IAIertaDAO = new JDBCAIertaDAO() ISG3 2010-2011 aler_id: String recuperarTodasLasAlertas() : List<Alerta> regi horainicio: Strin selectAlertaBvAlerid(int) : Alerta aler_nombrecontacto: String + registrarAlerta(Alerta): void regi_id: String muni_muniid: int regi_totalhoras: int selectAlertaBvContacto(String): Alerta prov_provid: int usua_usuaid: String serv_servid: int tibu_tibuid: int tise_tiseid: int tibu id: String Demandante Perfil Store dao: JDBCDemandanteDAO = new JDBCDemanda. dema_cod: String dao: JDBCPerfilDAO = new JDBCPerfilDAO dema_id: String addAlerta(String, String): void usua_usuaid: Strin consultarMisdatos(String) : Usuario_Registrado recuperarTodosLosPerfiles(): List<Perfil> selectPerfDescricoByPerfid(int): String listaServicioBvDemId(int) : List<Servicio> registarDemandante(): void perf descrico: String propone perf_descrila: String Usuario Registrado Enlace dao: JDBCRegistroDAO = new JDBCRegistr prov_provid: int nla descrico: Strinusua_clave: String Usuario Store usua_email: String recuperan odosLosRegistros(): List<Registro enla_id: String recuperarTodosLosReaistrosByUsuaid(int) : List<Registro> userDAO: JDBCUsuarioDAO usua usuario: String enla_url: String recuperarTookdemanda.trosByVolucod(String): List<Registro selectTotalHorasByVolucod(String) : int actualizarUsuario(Usuario_Registrado) : void borrarUsuario(int) : void 0..- 0..comprobarUsuario(String, String) : boolean insertarUsuario(Usuario Registrado): void isAdmin(Usuario_Registrado) : boolean isLogado(String, String) : boolean usua_usuaid: String isRegistrado(Connection, Usuario Registrado) : boolean dao: JDBCEnlaceDAO = new JDBCEnlaceDAO UAdministrador obtenerUsuald(String, String) : Integer recuperarIdByUsuarioClave(String, String): Integer recuperarTodosLosEnlaces(): \List<Enlace> recuperarUsuarioByldUsuario(String): Usuario_Registrat selectEnlaDescricoByEnlaid(int); String selectAllUsers(): List<Usuario> selectURLByEnlaid(int) : String selectUserById(String): Usuario_Registrado singOut(): boolean prov id: String prov_nombre: Strin Voluntario Store tise tiseid: String dao: JDBCVoluntarioDAO = new JDBCVolunta. vdao: IVoluntarioDAO = new JDBCVolunta muni nombre: Strine Provincia Store addAlerta(String, String) : void dao: JDBCProvinciaDAO = new JDBCProvinc consultarMisdatos(String): Usuario_Registrado obtenerMisdatos(String, String): Usuario listaServicioBvVolld(int) : List<Servicio> registrarVoluntario(Usuario): void recuperarTodasLasProvincias(): List<Provincia: registarVoluntario(): void selectProviciaByProvid(int) : String Tipo Servicio Store Tipo Servicio dao: JDBCMunicipioDAO = new JDBCMunicip. dao: JDBCTipoServicioDAO = new JDBCTipoSer.. tise_descrila: String Servicio Store recuperarTodosLosMunicipios(): List<Municipio recuperarTodosLosTipoServicio(): List<TipoServicio> selectMunicipioByMuniid(int): String selectTiseDescricoByServid(int): String dao: JDBCServicioDAO = new JDBCServici. selectProvinciaByMuniid(int) : String selectTiseDescricoByTiseid(int) : String actualizarServicio(Servicio): void borrarServicio(int): void insertarServicio(Servicio): void recuperarTodosLosServicios(): List<Servicio selectServDescricoByServid(int) : String selectServDescrilaByServid(int) : String selectTipoServicioByServid(int): String







DAO





Está Vd. en: INICIO - Menú Inicial

Usuario: cpompa

Ayuda

yuda :













Fin de la Presentación

Autores: Macarena Rodríguez Rodríguez Carlos Pompa Domínguez

