TEC | Tecnológico de Costa Rica

Trabajo Final de Graduación para optar por el título

Bachiller en Ingeniería en Computación

"Service Desk Interfaz Software"

Elaborado por Abraham Carvajal Vargas.

Carrera Ingeniería en Computación

Instituto Tecnológico de Costa Rica

Prof. Asesor: Oscar Víquez Acuña.

Sede San Carlos

04 Octubre del 2010

Resumen Ejecutivo

El proyecto “Service Desk” es desarrollado en Interfaz Software, empresa ubicada en San Ramón de Alajuela, la cual cuenta con varios años de experiencia en el mercado. Sin embargo se cuenta con la problemática de que no posee una herramienta que le permita clientes exponer una solicitud de atención para el soporte de posibles incidencias que presenten los productos realizados por la empresa.

A pesar de que nunca se han realizado desarrollos utilizando Ruby on Rails, el proyecto fue desarrollado en esta tecnología, la cual permite el desarrollo de forma muy rápida y fácil.

Este sistema pretende ofrecer a los clientes de Interfaz Software un medio por el cual puedan exponer una solicitud de atención ante la empresa, para recibir la atención deseada de forma más simple y eficaz. Además permite a la empresa un mejor control sobre la atención que reciben los clientes.

La arquitectura presente en este proyecto es Modelo-Vista-Controlador, la cual divide en campas la estructura de una aplicación, permitiendo un mejor manejo y control durante el desarrollo, permitiendo así tener un control mayor de la aplicación.

Parte del éxito en este proyecto son las facilidades que brinda el Framework Rails, que permite a partir de comandos generar de manera automática plantillas las cuales son reflejos de objetos de Base de Datos y no hace falta establecer conexión entre ellas.

También se cuenta con la facilidad de gemas que son el equivalente a plug-ins, componentes, servicios que realizan una cantidad significativa de funciones que resultan muy útiles, ya que facilitan muchas funciones gracias a la reutilización de componentes.

Tabla de contenido

1. Tabla de figuras 4
2 Contexto del proyecto 5
2.1 Empresa 5
2.2 Antecedentes del proyecto 6
3 Descripción del problema 7
3.1 Perfil de los interesados 8
3.2 Necesidades y expectativas 9
3.3 Supuestos y dependencias 10
3.4 Requerimientos no funcionales 10
3.5 Características generales 10
4 Objetivos y Alcances del sistema 11
4.1 Objetivo general 11
4.2 Objetivo específicos 11
4.3 Funciones del sistema 12
5. Modelo de Diseño 13
5.1 Arquitectura conceptual de la solución 13
5.2 Los modelos de subsistemas 14
5.3 Diagrama de clases 15
5.4 Interfaces de usuario 21
5.4.1 Plantillas Base 22
5.4.2 Aplicación “Service Desk” 25
5.5 Componentes y servicios 27
5.6 Diseño base datos 29
6 Conclusiones y comentarios 30
7 Referencias 32

1. Tabla de figuras

Ilustración 1 - Organigrama 5
Ilustración 2 - Modelo Vista Controlador 13
Ilustración 3 – Diagrama de Clases 15
Ilustración 4 – Modelo Entidad-Relación 29

2 Contexto del proyecto

2.1 Empresa

Interfaz Software fue fundada en el año 2002 con amplia experiencia en el desarrollo de software para internet, es parte de Grupo Interfaz, empresa la cual se divide en 3 departamentos, pero siempre siendo empresas hermanas.

Cuenta con equipo de profesionales especializados en diseño, desarrollo de sitios web y aplicaciones para internet en constante investigación, experimentación con tecnologías de punta, tendencias y mejores prácticas con el objetivo de brindarle a cada cliente una solución ideal para cada necesidad, nueva o adaptada a su plataforma tecnológica actual.

La flexibilidad de sus soluciones y procesos de desarrollo ha permitido establecer relaciones de largo plazo con importantes compañías en el sector financiero, industria y servicios.

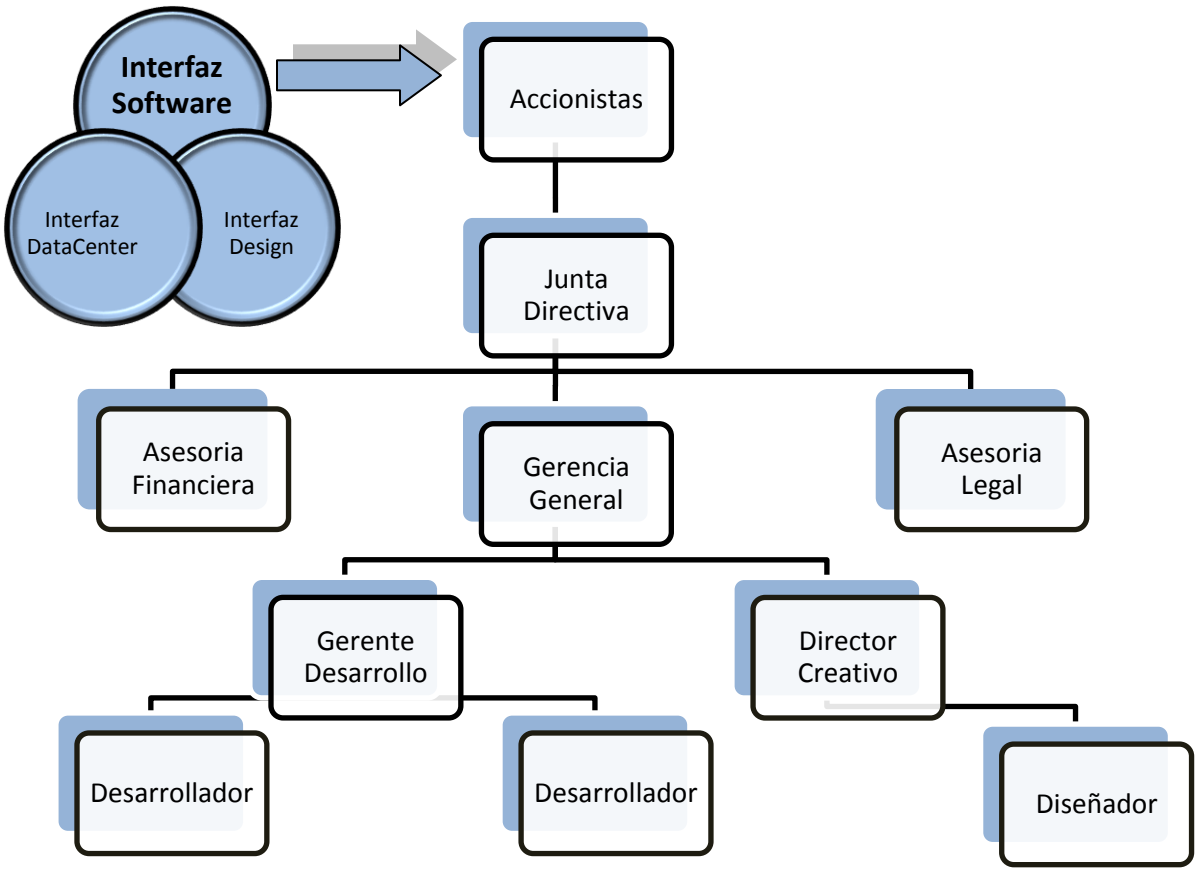


Ilustración 1 - Organigrama

2.2 Antecedentes del proyecto

Previamente se quiso realizar un proyecto similar a este en la empresa utilizando Java en la solución pero debido a ciertos factores no se pudo completar en su totalidad, dentro de estos los más significativos fueron la inexperiencia junto con la falta de tiempo.

La falta de tiempo e inexperiencia se convirtieron en factores cruciales que afectaron la culminación exitosa de ese proyecto, debido que estaba siendo desarrollado como requisito de un curso por estudiantes, el que se enfocaba más el área de documentación por lo tanto nunca se produjo producto final, razón por la cual se decidió implementar algo nuevo utilizando otras tecnologías más de corte novedoso.

Actualmente se comenzará desde cero utilizando otro tipo de tecnología, Ruby on Rails, abarcando el proceso de Planeación, Diseño, Desarrollo, Implementación y Validación; sin dejar de lado la Investigación, que dadas las características de este proyecto se realizara en paralelo durante la elaboración de todos los módulos correspondientes.

3 Descripción del problema

La empresa cuenta con la problemática de que no cuenta con una herramienta que permita a los clientes exponer una solicitud de atención para el soporte de posibles incidencias que presenten los productos realizados por la empresa, el proceso actual requiere de muchas llamadas y correos lo cual genera perdida de información de casos previos junto con la existencia de problemas en cuanto a asignación de recursos y difícil control de solicitudes.

La solución que se plantea a este problema es una aplicación en ambiente web tipo Service Desk, que ayude a gestionar y solucionar los problemas que se le presenten al cliente, automatizando todo el proceso por parte de ambas partes, tanto cliente como empresa, de esta manera se espera manejar un flujo significativo de información de forma efectiva, facilitando la generación de reportes y manejo de problemas, todo esto usando Ruby on Rails.

3.1 Perfil de los interesados

Stakeholder	Responsabilidades	Intereses
Adrian Hernández / Director Creativo	Participar en la conceptualización del proyecto. Obtener y consolidar las necesidades de la empresa. Aprobar especificación de requerimientos y prototipo. Realizar pruebas de aceptación del proyecto.	Que el proyecto inicie en el II semestre del año. Mayor atractivo para clientes. Seguimiento por parte del cliente a sus casos. Control interno.
Esteban Calderón Ramírez / Gerente Desarrollo	Aclarar dudas de las áreas de desarrollo. Aprobar especificación de requerimientos y prototipo. Llenar encuesta de satisfacción. Revisar y aprobar documentación técnica relevante para el análisis y diseño del proyecto. Revisar estandarización del código. Realizar pruebas de aceptación del proyecto.	Que el proceso de desarrollo siga un estándar aceptable tanto de diseño como el desarrollo como tal.
Abraham Carvajal / Departamento Desarrollo	Analista/Desarrollador	Diseñar y desarrollar el producto.

3.2 Necesidades y expectativas

Necesidades / Expectativas	Problema	Solución actual	Solución propuesta
Sistematizar la atención a clientes de Interfaz, tanto en la solución de incidentes en productos entregados como para nuevas solicitudes. (Prioridad alta)	Difícil atención de clientes a la hora de solicitar soporte en sus productos.	Proceso realizado mediante llamadas telefónicas, vía correo electrónico.	Sistema sea el encargado de recoger toda esta información y crear notificaciones electrónicas correspondientes.
Gestionar de manera automatizada posibles incidencias y solicitudes de parte de los clientes. (Prioridad alta)	Dificultad a la hora de atender las solicitudes de los clientes.	Proceso realizado mediante llamadas telefónicas, vía correo electrónico.	Sistema agrupe las solicitudes correspondientes y a partir de estas crear tickets para solucionar dicho caso.
Gestionar la asignación de recurso humano. (Prioridad alta)	Conflicto en la asignación de recurso humano.	Asignación realizada de acuerdo a carga de trabajo actual.	Sistema asigne a recurso humano carga equitativa de acuerdo a situación actual.
Facilitar la comunicación empresa-cliente. (Prioridad alta)	Complicada atención entre empresa-cliente, no ocurre con fluidez.	Comunicación realizada por correo, teléfono y mediante acceso remoto.	Notificaciones electrónicas las cuales mantienen informadas a ambas partes acerca de la situación actual.

3.3 Supuestos y dependencias

Se debe contar con una versión Rails > 2.X

Se debe contar con las gemas: Paperclip Gem (manejo adjunto), Will_paginate Gem (manejo paginación), STMP server (envió correos), además ImageMagic (necesario para Paperclip Gem)

3.4 Requerimientos no funcionales

- La solución debe estar diseñada y desarrollada Ruby on Rails.
- Basada en web.
- MySQL como motor de Base de datos.
- Basada en arquitectura MVC.

3.5 Características generales

- Mejor comunicación entre la empresa y clientes para aumentar la calidad a la hora de atención de solicitudes.
- Mayor control sobre el soporte que brinda la empresa a sus clientes para obtener información relevante de manera rápida.
- Mejorar la imagen y confianza de la empresa para tener a los clientes satisfechos, ser más atractivos en el mercado.

4 Objetivos y Alcances del sistema

4.1 Objetivo general

Sistematizar la atención a clientes de Interfaz Software S.A, tanto en la solución de incidentes en productos entregados como para nuevas solicitudes.

4.2 Objetivo específicos

- Gestionar y solucionar posible incidencias de los productos (software).
- Permitir a los clientes efectuar solicitudes de nuevos proyectos.
- Comunicar de forma sencilla y eficiente a la empresa con los clientes.

Este sistema les ofrece a los clientes de Interfaz Software un medio en el que puedan exponer una solicitud de atención ante la empresa, para recibir el soporte deseado de forma más sencilla y eficiente. Además permite a la empresa un mejor control sobre la atención que reciben los clientes.

4.3 Funciones del sistema

Función	Prioridad	Descripción
Administrar empresas	Alta	Proceso mediante el cual se agrega una empresa, añadiendo los datos correspondientes, también se podrán administrar teléfonos y aéreas al mismo tiempo, sin dejar de lado quien será el encargado de la misma
Administrar personas	Alta	Proceso mediante el cual se agrega una persona, añadiendo los datos correspondientes, también se podrán administrar teléfonos y correos al mismo tiempo.
Administrar proyectos	Alta	Proceso mediante el cual se agrega proyecto
Administrar solicitud	Alta	Proceso mediante el cual se agrega solicitud
Administrar ticket	Alta	Proceso mediante el cual se agrega ticket, seleccionando los debidos datos durante la asignación, se podrán realizar comentarios.
Búsquedas	Alta	Proceso mediante el cual se podrán realizar búsquedas de acuerdo a distintas categorías, estos a nivel de solicitudes y tickets.
Autenticación de usuarios	Alta	Sistema autentifica a usuarios y de acuerdo al tipo de estos, muestra distinta pantalla.

5. Modelo de Diseño

5.1 Arquitectura conceptual de la solución

La arquitectura que presenta el proyecto es MVC [1] (modelo-vista-controlador) en donde:

- Modelo es todo acceso a base de datos, y las funciones que contienen la "lógica de negocio".
- La vista se encarga de mostrar la información al usuario final: HTML, XML.
- El controlador une la vista con el modelo, contiene toda la lógica de programación. Almacena las funciones que toman los valores de un formulario, delega consultas de base de datos al modelo y produce valores que invocarán a la vista adecuada.

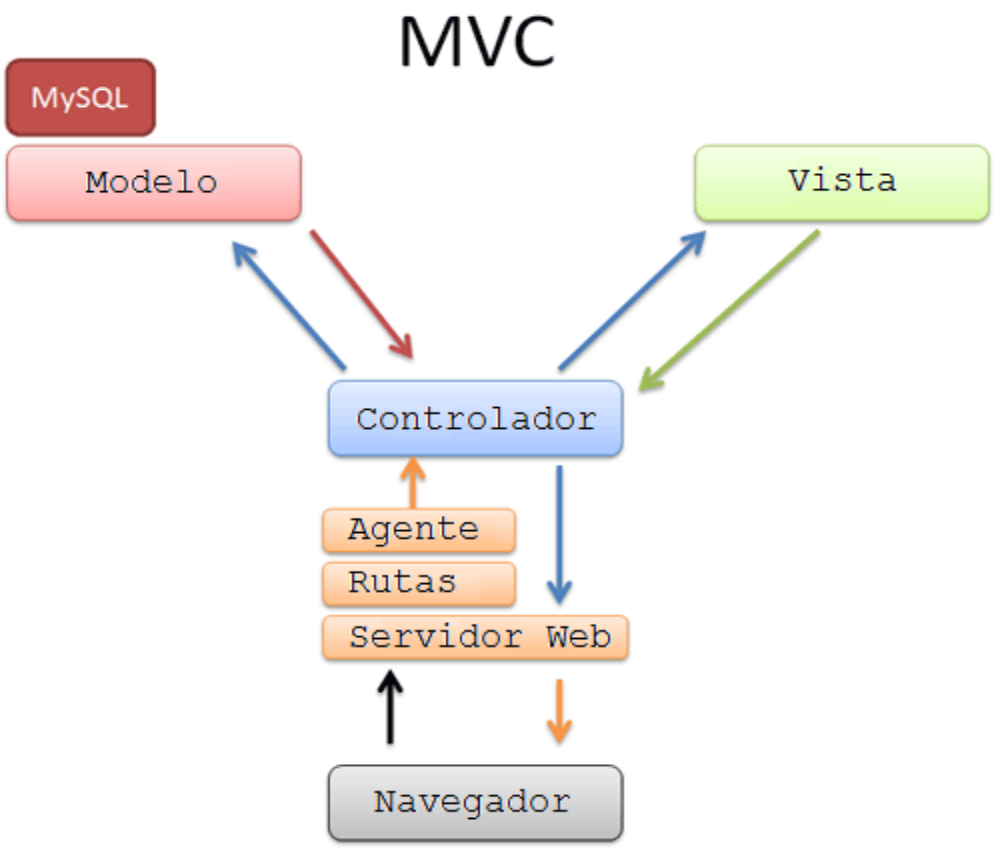
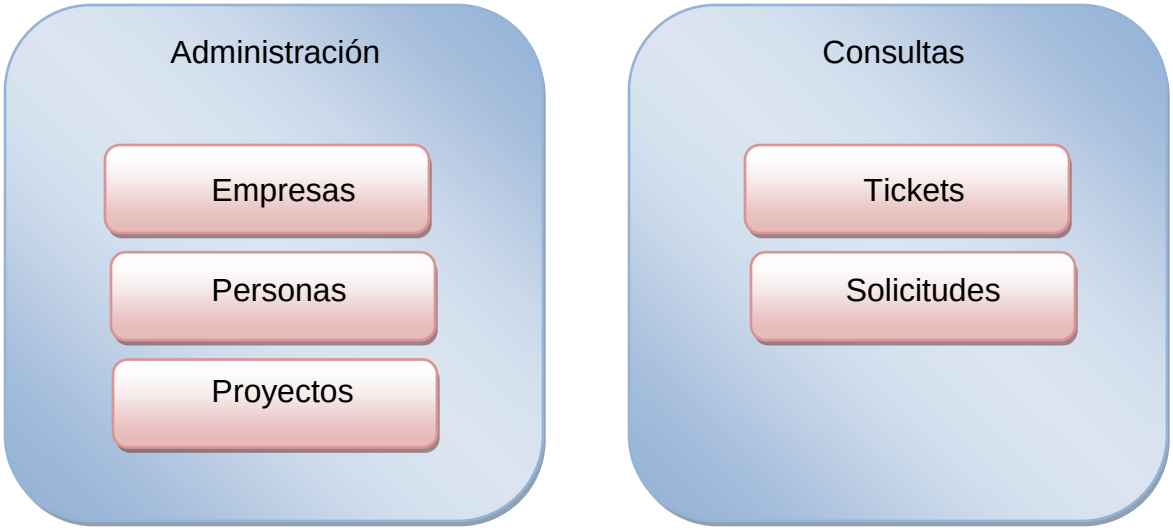


Ilustración 2 - Modelo Vista Controlador

5.2 Los modelos de subsistemas

Es importante recalcar que la aplicación posee varios tipos de usuarios: administrador, operador, cliente, debido a esto los modelos se agrupan en dos grandes módulos los cuales son Administración y Consultas.



Administración:

- Empresas: permite agregar, editar y buscar empresas por diferentes criterios.
- Personas: permite agregar, editar y buscar personas por diferentes criterios.
- Proyectos: permite agregar, editar y buscar proyectos por diferentes criterios.

Consultas:

- Solicitudes: permite agregar, editar (solo administrador) y buscar solicitudes por diferentes criterios, agrupa las asignadas a cada usuario.
- Tickets: permite agregar (solo administrador), editar y buscar solicitudes por diferentes criterios, agrupa las asignadas a cada usuario.

En cuanto al módulo de administración es accesible únicamente por administrador.

5.3 Diagrama de clases

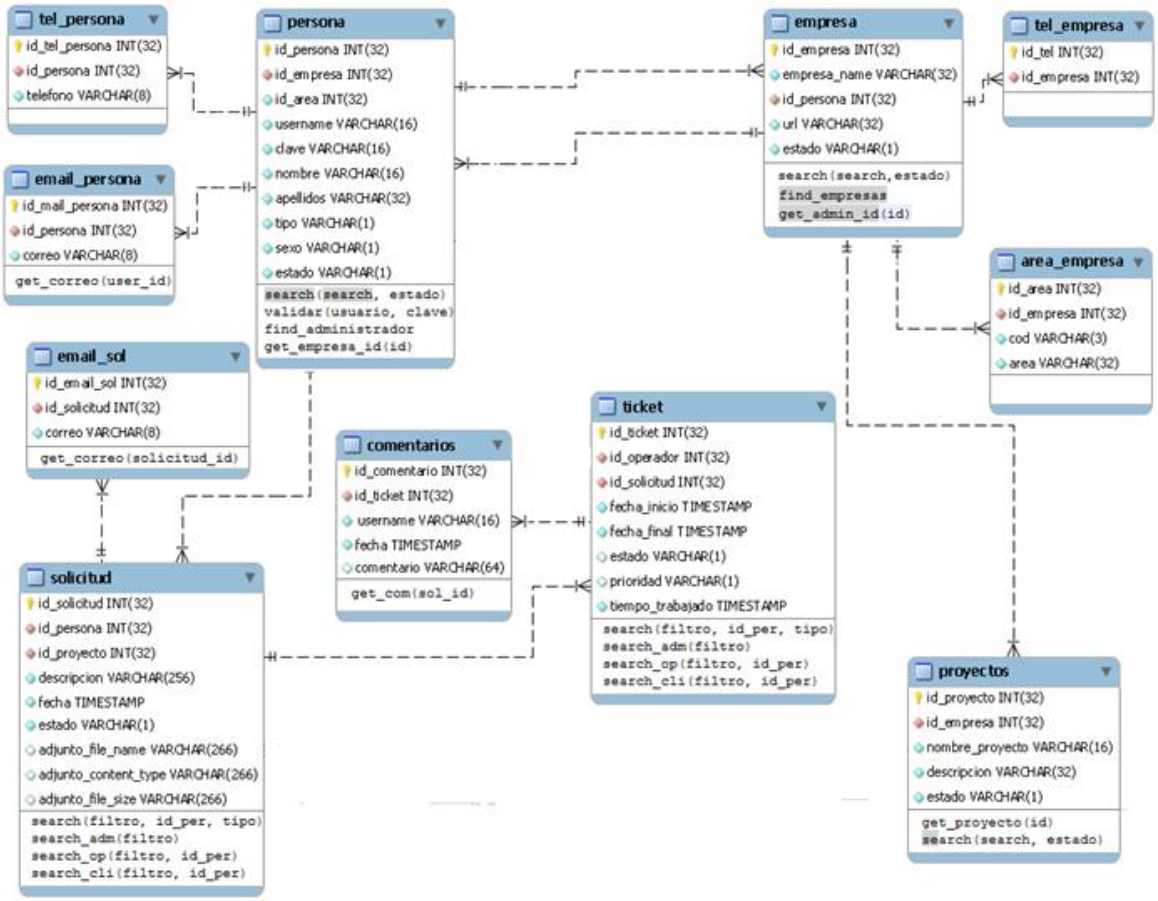


Ilustración 3 – Diagrama de Clases

Empresa	
Atributos	• `id_empresa` INTEGER(32) UNSIGNED AUTO_INCREMENT, • `empresa_name` VARCHAR(32) not null unique, • `id_persona` INTEGER(32) UNSIGNED not null, • `url` VARCHAR(32) not null unique, • `estado` VARCHAR(1) not null, /* a(activa) - i(inactiva) - */
Relaciones	• Posee varios tel_empresa. • Posee varias areas_empresa. • Posee varias personas. • Posee una persona.
Métodos	• Search: busca la empresa de acuerdo a filtro o por defecto, lo devuelve en variable de instancia. • Find_empresa: devuelve la empresa perteneciente a una persona. • Get_admin_id: devuelve los administradores de la aplicación.

Tel_Empresa	
Atributos	• `id_tel` INTEGER(32) UNSIGNED AUTO_INCREMENT, • `id_empresa` INTEGER(32) UNSIGNED not null, • `telefono` VARCHAR(8),
Relaciones	• Pertenece a empresa
Métodos	

Area_Empresa	
Atributos	• `id_area` INTEGER(32) UNSIGNED AUTO_INCREMENT, • `id_empresa` INTEGER(32) UNSIGNED not null, • `cod` VARCHAR(3) not null, • `area` VARCHAR(32)not null,
Relaciones	• Pertenece a empresa
Métodos	

Persona	
Atributos	• `id_persona` INTEGER(32) UNSIGNED AUTO_INCREMENT, • `id_empresa` INTEGER(32) UNSIGNED not null, • `id_area` INTEGER(32) UNSIGNED not null, • `username` VARCHAR(16) not null unique, • `clave` VARCHAR(16) not null, • `nombre` VARCHAR(16) not null, • `apellidos` VARCHAR(32) not null, • `tipo` VARCHAR(1) not null, /* o(operador) - a(administrador) - c(cliente) */ • `sexo` VARCHAR(1) not null, /* m(masculino) - f(femenino) */ • `estado` VARCHAR(1) not null, /* a(activo) - i(inactivo) */
Relaciones	• Pertenece a empresa • Posee varios telefonos_persona. • Posee varios email_persona.
Métodos	• Search: busca la persona de acuerdo a filtro o por defecto, lo devuelve en variable de instancia. • Validar: valida el acceso a con usuario y contraseña. • Find_administrador: devuelve el administrador de acuerdo a empresa relacionada con persona. • Get_empresa_id: devuelve el id de la empresa relacionada con persona.

Tel_Persona	
Atributos	• `id_tel_persona` INTEGER(32) UNSIGNED AUTO_INCREMENT, • `id_persona` INTEGER(32) UNSIGNED not null, • `telefono` VARCHAR(8) not null,
Relaciones	• Pertenece a persona.
Métodos	

Email_Persona	
Atributos	• `id_mail_persona` INTEGER(32) UNSIGNED AUTO_INCREMENT, • `id_persona` INTEGER(32) UNSIGNED not null, • `correo` VARCHAR(8) not null,
Relaciones	• Pertenece a persona.
Métodos	• Get_correo: devuelve los correo perteneciente a persona

Solicitud	
Atributos	• `id_solicitud` INTEGER(32) UNSIGNED AUTO_INCREMENT, • `id_persona` INTEGER(32) UNSIGNED not null, • `id_proyecto` INTEGER(32) UNSIGNED not null, • `descripcion` VARCHAR(256) not null, • `fecha` TIMESTAMP not null, • `estado` VARCHAR(1) not null, /* e(espera) - a(atendido) - f(finalizado) */ • `adjunto_file_name` VARCHAR(266) null, • `adjunto_content_type` VARCHAR(266) null, • `adjunto_file_size` VARCHAR(266) null,
Relaciones	• Pertenece a persona. • Pertenece a proyecto. • Posee un ticket. • Posee varios correo_sol.
Métodos	• Search: busca las solicitudes de acuerdo a filtro o por defecto, lo devuelve en variable de instancia. • Search_adm: busca las solicitudes para administrador. • Search_op: busca las solicitudes para operador. • Search_cli: busca las solicitudes para cliente.

Email_sol	
Atributos	• `id_email_sol` INTEGER(32) UNSIGNED AUTO_INCREMENT, • `id_solicitud` INTEGER(32) UNSIGNED not null, • `correo` VARCHAR(8) not null,
Relaciones	• Pertenece a solicitud.
Métodos	• Get_correo: devuelve los correos pertenecientes a una solicitud.

Proyecto	
Atributos	• `id_proyecto` INTEGER(32) UNSIGNED AUTO_INCREMENT, • `id_empresa` INTEGER(32) UNSIGNED not null, • `nombre_proyecto` VARCHAR(16) not null, • `descripcion` VARCHAR(32) not null, • `estado` VARCHAR(1) not null, /* e(espera) - a(atendido) - f(finalizado) */
Relaciones	• Pertenece a empresa. • Posee varias solicitudes.
Métodos	• Search: busca los proyectos de acuerdo a filtro o por defecto, lo devuelve en variable de instancia. • Get_proyecto: devuelve el proyecto perteneciente a empresa.

Ticket	
Atributos	• `id_ticket` INTEGER(32) UNSIGNED AUTO_INCREMENT, • `id_operador` INTEGER(32) UNSIGNED not null, • `id_solicitud` INTEGER(32) UNSIGNED not null, • `fecha_inicio` TIMESTAMP not null, • `fecha_final` TIMESTAMP not null, • `estado` VARCHAR(1), /* a(atendido) - d(devuelto) - g(reasignado) - r(reabierto) - x(finalizado por operador) - y(finalizado por cliente) - z(cerrado) */ • `prioridad` VARCHAR(1), /* a(alta) - m(media) - b(baja) */ • `tiempo_trabajado` timestamp not null,
Relaciones	• Pertenece a solicitud. • Posee varios comentarios.
Métodos	• Search: busca las tickets de acuerdo a filtro o por defecto, lo devuelve en variable de instancia. • Search_adm: busca las tickets para administrador. • Search_op: busca las tickets para operador. • Search_cli: busca las tickets para cliente.

Comentario	
Atributos	• `id_comentario` INTEGER(32) UNSIGNED AUTO_INCREMENT, • `id_ticket` INTEGER(32) UNSIGNED not null, • `username` VARCHAR(16) not null, • `fecha` TIMESTAMP not null,
Relaciones	• Pertenece a ticket.
Métodos	• Get_comentario: devuelve el id_ticket de un determinado comentario.

5.4 Interfaces de usuario

Al ser una aplicación en ambiente WEB se manejan una serie de plantillas, en su mayoría agrupadas de acuerdo a acción, característico del Framework Rails, permitiendo genericidad a partir de objetos de la base de datos.

Esto se logra con la funcionalidad de Scaffold incluida en Rails, que con solo unos cuantos comandos se genera el modelo, vista, controlador y permite crear la tabla en la base de datos. Además, siguiendo la teoría de CRUD [2], el comando de Scaffold genera también toda la gestión para crear, eliminar, editar y mostrar los datos del modelo en que se está trabajando, por lo que se tiene las funciones básicas de un mantenimiento de una tabla.

El Framework permite la elaboración de aplicaciones de forma rápida, siendo muy útil y ahorrando mucho tiempo, gracias a que los componentes están integrados y no hace falta establecer conexión entre ellos, permitiendo la administración de la aplicación desde un alto nivel el cual facilita las tareas para el programador, creando la facilidad para realización de código de forma rápida y con menos errores, haciendo que el código sea fácil de entender, mantener y mejorar. [3]

Ejemplo:

```
script/generate scaffold persona nombre:string apellidos:string
rake db:migrate
```

5.4.1 Plantillas Base

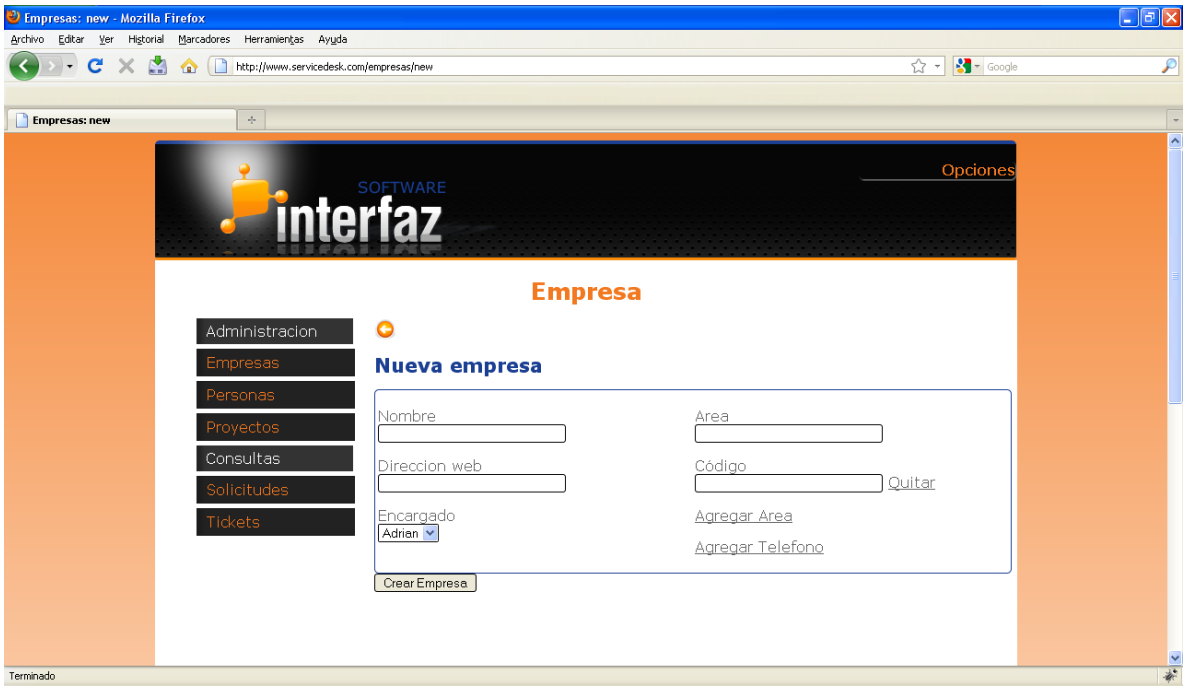
Plantilla General

Plantilla Index

Plantillas nuevo, editar, mostrar

5.4.2 Aplicación "Service Desk"

Seguidamente se muestra el resultado de la aplicación de las plantillas en el proyecto de creación del Service Desk WEB para la empresa Interfaz Software. Cabe resaltar los detalles de implementación similares a las plantillas presentadas en el apartado anterior.


Empresas: new - Mozilla Firefox
Archivo Editar Ver Historial Marcadores Herramientas Ayuda
http://www.servicedesk.com/empresas/new
Empresas: new
SOFTWARE
interfaz
Opciones
Empresa
Administracion
Empresas
Personas
Proyectos
Consultas
Solicitudes
Tickets
Nueva empresa
Nombre
Area
Direccion web
Código
Quitar
Encargado
Adrian
Agregar Area
Agregar Telefono
Crear Empresa
Terminado


Personas: index - Mozilla Firefox
Archivo Editar Ver Historial Marcadores Herramientas Ayuda
http://www.servicedesk.com/personas
Personas: index
SOFTWARE
interfaz
Opciones
Personas
Todos
Buscar
Administracion
Empresas
Personas
Proyectos
Consultas
Solicitudes
Tickets
Persona
Adrian Hernandez
cliente sd
operador sd
Terminado

5.5 Componentes y servicios

Seguidamente se explica de manera general los principales componentes usados en la estructura general de la aplicación.

Active record:

Active Record es un enfoque al problema de acceder a los datos de una base de datos. Una fila en la tabla de la base de datos (o vista) se envuelve en una clase, de manera que se asocian filas únicas de la base de datos con objetos del lenguaje de programación usado. Cuando se crea uno de estos objetos, se añade una fila a la tabla de la base de datos. Cuando se modifican los atributos del objeto, se actualiza la fila de la base de datos. La clase envoltorio implementa métodos de acceso para cada columna de la tabla o vista. De esta manera se vuelve relativamente fácil operaciones de CRUD en la aplicación porque permite mapear la base de datos en objetos.

Atreves de Scaffold con unos cuantos comandos se consigue tener las operaciones CRUD de forma rápida y efectiva.

Ejemplo:

```
script/generate scaffold persona nombre:string apellidos:string
rake db:migrate
```

Con esos comandos se crea objeto persona en base de datos, donde se podrá acceder mediante http://localhost:3000/persona/[new|edit|index|destroy] donde persona será el controlador y new, edit, index, destroy la acción que estará ligada a una vista. [4]

Paper-clip:

Es una gema para Ruby on Rails Active Record que convierte archivos en atributos de la misma manera que los campos de una base de datos normal, permitiendo la manipulación de archivos de la misma manera que Active Record extendiendo todos sus métodos.

Ejemplo:

```
has_attached_file :adjunto,
:url  => "/assets/solicituds/:id/:style/:basename.:extension",
:path => ":rails_root/public/assets/:id:basename.:extension"
```

Con esto se modifica la base de datos y se agrega columnas al objeto referenciado sobre un modelo, mediante el nombre adjunto/campo, donde campo puede ser tipo, tamaño, nombre, ruta, permitiendo la manipulación de archivos. [5]

Will-paginate:

Es una gema para Ruby on Rails Active Record que proporciona una paginación simple, potente y amplio. También la prestación de enlaces de paginación en plantillas ActionView.

Ejemplo:

Controlador:

```
def index
    @personas = Persona.search(params[:search],
params[:estado]).paginate(:per_page => 20, :page => params[:page])


    respond_to do |format|
      format.html # index.html.erb
      format.js   # index.js.erb
      format.xml  { render :xml => @personas }
    end
  end
```

Vista:

```
<%= will_paginate @persona %>
```

Con esto ya paginará los resultados para la acción index del objeto persona. [6]

Mongrel_service:

Es una gema para Ruby on Rails que permite la creación de un servicio de Windows para el inicio del Servidor Mongrel quien es el encargado de operar las aplicaciones Rails.

Ejemplo:

```
mongrel_rails     service::install     -N     AplicacionRails     -c
RutaDeLaAplicacion -p 4001 -e production
```

Con esto se crea el servicio de Windows y mediante un servidor web se puede re direccionar tráfico a este. [7]

5.6 Diseño base datos

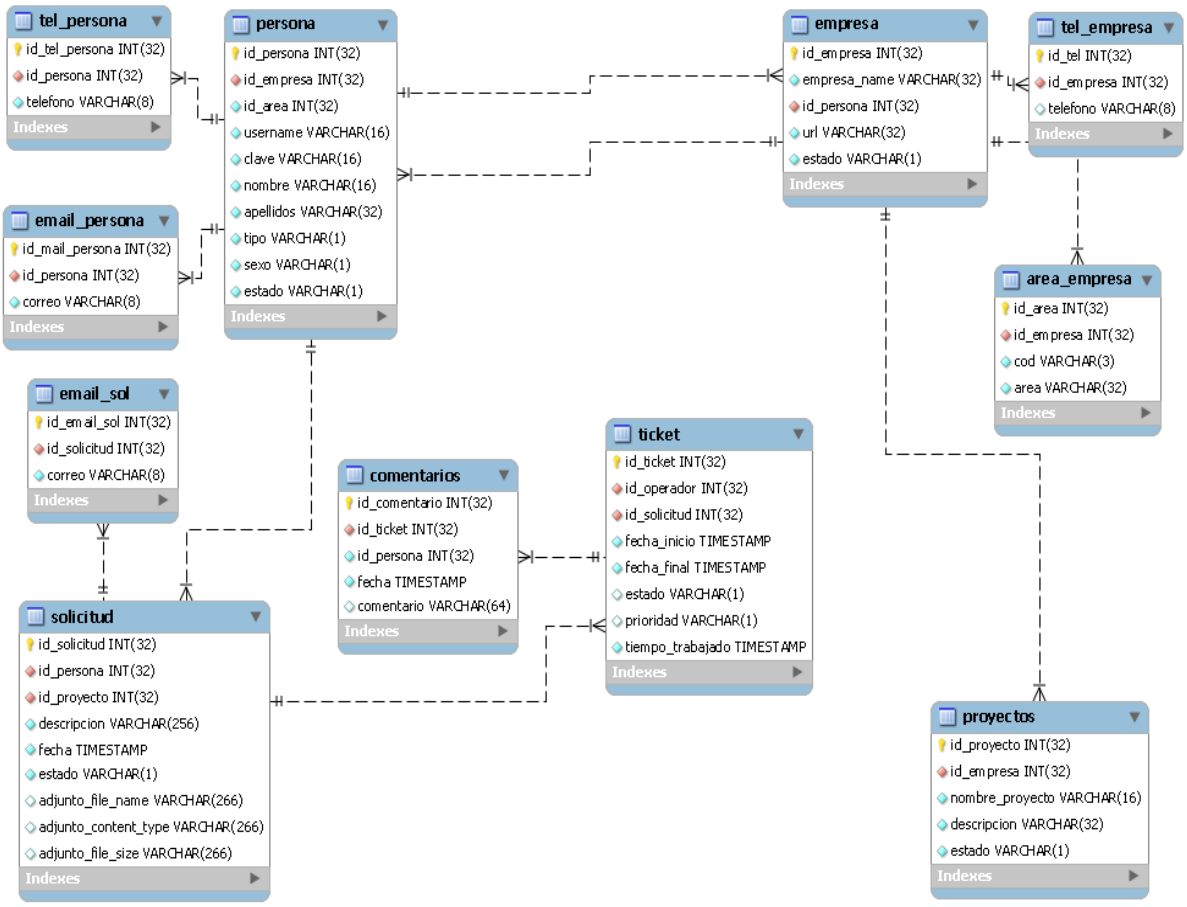


Ilustración 4 – Modelo Entidad-Relación

6 Conclusiones y comentarios

Una de las claves para el éxito en el desarrollo y por tanto la implementación completa del sistema Service Desk fue la utilización del Framework Ruby on Rails que permitió el desarrollo de forma rápida y sencilla, utilizando una analogía es como recibir un juego de legos en donde lo único que se debe hacer es armar un componente, sin tener preocuparse si las piezas calzan entre sí, cual es la estructura que debería tener cada una, el tamaño y forma de estas, es un mundo en donde no se debe repetir lo mismo una y otra vez.

Las gemas de Ruby proporcionan una fuente de elementos, los cuales facilitaron el desarrollo de ciertos componentes en la aplicación, esto permite tener una cantidad significativa de componentes con funcionalidades increíbles sin costo adicional, con la característica de software libre, dispersas en la red o en el sitio web oficial de gemas de Ruby, son muy fácil de instalar y utilizar.

La formación recibida en el proceso universitario es útil cuando se necesitan conocimientos técnicos, pero sin embargo la vida laboral es otro mundo, es distinto el desarrollo de un proyecto en el cual no se espera una nota, en donde ya hay reputación y responsabilidad de por medio, la experiencia de estar en una empresas es gratificante.

La manipulación de base de datos es muy interesante porque se manejan a nivel lógico todas las relaciones entre tablas, permitiendo manipular todo esto desde la capa de modelo, de forma sencilla y eficiente, características de tecnologías de este tipo, permitiendo validaciones a la hora de guardar datos de manera realmente simple, sin complicaciones.

La portabilidad que posee el Framework es increíble, reflejo de esto es que para cambiar el motor de base de datos basta con cambiar una única línea de un archivo de configuración, sin tener que hacer una nueva conexión, sin repetir lo mismo una y otra vez.

Aplicaciones de tipo Rails corren prácticamente sobre cualquier servidor web y sobre sistemas operativos más comunes, sin tener que preocuparse por cuestiones de arquitectura y elementos de este tipo, ventaja que hace sumamente atractivo el desarrollo en esta tecnología.

La compatibilidad del Framework con elementos como Ajax, JQuerry, Prototype, CSS hace el ambiente de desarrollo sumamente agradable, debido a que casi nunca se debe hacer uso de elementos externos, se cuenta con todo lo necesario dentro del mismo ambiente.

La generación del esqueleto de la aplicación resulta realmente sencillo, basta un par de comandos para generar desde la base de datos hasta el controlador, modelo y vista de un objeto en el peor de los escenarios, porque algunas herramientas ya poseen estas opciones a partir de un par de clics.

Ruby on Rails permite desarrollar en tres ambientes distintos de forma paralela, prueba, desarrollo y producción, junto con la consola interactiva de Rails la cual permite hacer un sin número de pruebas de forma local sin necesidad de tener que depender de elementos adicionales a la aplicación.

Es importante resaltar que la documentación de Ruby on Rails en su mayoría está en inglés, podría considerarse como una barrera para los que quisieran comenzar a desarrollar desde cero, aunque en su mayoría este en inglés, hay una cantidad realmente significativa en la web.

7 Referencias

[1] Exequiel, Catalani. ARQUITECTURA Modelo/Vista/Controlador, Consultado el 13 octubre 2010, de http://exequielc.wordpress.com/2007/08/20/arquitectura-modelovistacontrolador/

[2] Dominic, Da Silva. Skip the CRUD: Building a Simple Database Backed Web Application Using Ruby on Rails, Consultado el 13 octubre 2010, de http://www.developer.com/db/article.php/3656831/Skip-the-CRUD-Building-a-Simple-Database-Backed-Web-Application-Using-Ruby-on-Rails.htm

[3] Ariel, Orozco. Ruby On Rails: Scaffold, Consultado el 13 de octubre 2010, de http://geekbass.blogspot.com/2008/08/ruby-on-rails-scaffold.html

[4] Active Record, Consultado el 6 de setiembre 2010, de http://api.rubyonrails.org/classes/ActiveRecord/Base.html

[5] RubyGems.org: Paperclip, Consultado el 6 de setiembre 2010, de http://github.com/thoughtbot/paperclip

[6] RubyGems.org: Will_paginate, Consultado el 6 de setiembre 2010, de http://github.com/mislav/will_paginate

[7] RubyGems.org: Mongrel_service, Consultado el 11 de octubre 2010, de http://github.com/fauna/mongrel_service