Descripción Arquitectónica

SOSEMADO: Software de Servicios de Mantenimiento a Domicilio

Desarrollo del Software 2020 - 2021 ETSIIT UGR

UNIVERSIDAD DE GRANADA

Ahmed El Moukhtari Koubaa
Iván Valero Rodríguez

Descripción del sistema

SOSEMADO es un Sistema Software para realizar tareas de Mantenimiento a Domicilio, donde un Cliente crea un trabajo, y ese trabajo se asigna a un Técnico.

Si este lo acepta, mandaría su presupuesto con un precio determinado. En caso contrario, se le reasigna a otro técnico, y así hasta que alguien lo elija o todos rechacen.
.
Para la selección del Técnico, se usan parámetros como la valoración de los clientes y su tipo de especialidad (albañilería, fontanería...)

Se habilitará un inicio de sesión y un registro para permitir el acceso privado a los trabajos por parte de técnicos y clientes del sistema.

Análisis de requisitos

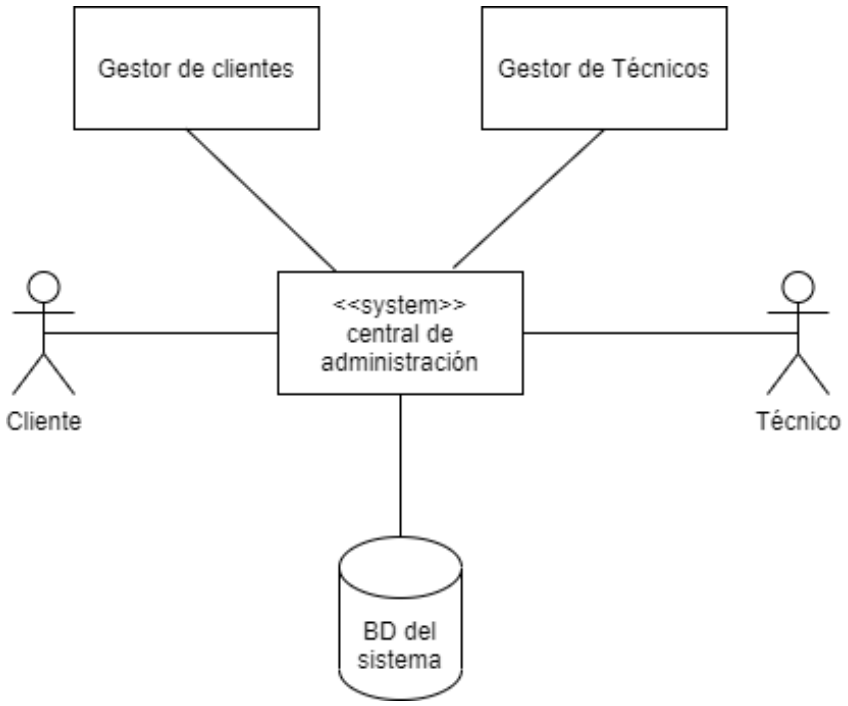
Requisitos funcionales	Requisitos no funcionales
- Dar alta cliente - Cliente define trabajo - Dar baja cliente - Modificar trabajo - Cancelar trabajo creado - Asignar trabajo dinámicamente	- Darse de alta es un proceso fácil y sencillo, toma menos de 5 minutos - Se mantendrá un registro de los trabajos - La interfaz del cliente estará hecha en Ruby on Rails y será accesible por web - La base de datos usada será relacional - La base de datos será MySQL

Los interesados y sus inquietudes

Partes interesadas	**Intereses**
Cliente (customer)	- Resolver su problema lo antes posible Encontrar un técnico de calidad

	- Una oferta competitiva
Técnico (realizadores de trabajos)	- Conseguir nuevos clientes - Tener presencia en el mundo digital - Beneficio económico - Asignación automática de trabajos
Administradores de empresa	- Beneficio económico (por publicidad) - Captar usuarios - Expandir la marca - Hacer contratos con agencias privadas - Expandirse ofreciendo nuevos servicios
Otras empresas del sector	- Expandirse a través de nuestra marca - Llegar a nuevos usuarios - Nuevas tecnologías para publicidad
Organizaciones públicas	- Beneficio económico (impuestos) - Colaboraciones con empresas (a través de nuestro sistema)

Diagrama de arquitectura del sistema



Criterios de calidad a partir de las distintas perspectivas

Seguridad

Una de las grandes inquietudes del sistema por parte, sobre todo, de los clientes, es la protección de la Información. De forma paralela, por la Ley Orgánica de Protección de Datos, se pide que se trate de proteger la información personal de los Clientes y Técnicos, ya que si la información se filtrara se podría llevar a vulnerar sus derechos (y causaría problemas legales).

Por ello,

- La contraseña se encriptará usando un algoritmo de hash como MD5, SHA… (si bien no debería dar perjuicio con respecto a la aplicación para técnicos)
- Debería darse la información justa y necesaria para realizar el Trabajo, o mostrar la información relevante de contacto.

Desempeño

El desempeño del sistema implica que se permite realizar las operaciones con relativa rapidez, ya que a los usuarios, tanto clientes como técnicos, no les interesa esperar. Por

ello, no debería tardarse más de 10 segundos en las operaciones más complejas, y sobre todo, cumplir con lo comentado en los requisitos no funcionales.

Disponibilidad

Otro de los grandes puntos del sistema a tener en cuenta de cara a realizar criterios de calidad es la disponibilidad, es decir, cómo tolerará los errores y cómo dejar que el sistema esté funcionando el mayor tiempo posible. Por ello:

- El sistema web se desplegará en el servidor ofrecido para realizar las prácticas.
- Se trabajará lo máximo posible en local antes de subirlo al servidor.

Evolución

Si bien el proyecto no daría para hacer una mayor evolución, se ha pensado en sistemas alternativos de asignación de técnicos a trabajos, como la subasta por el menor precio. En parte, también se trata del sistema que se ha ido ideando durante las anteriores prácticas.

Diseño de pruebas

Además de realizar el sistema, habrá que realizar una serie de pruebas. Por nuestra parte, se recomendaría probar los siguientes puntos, si bien los del equipo adyacente podrán realizar más o menos según vean convenientes:

Unitarias

Comprobar que si el cliente define un trabajo, el atributo `Cliente.trabajo` no esté nulo.

Comprobar que si al trabajo se le asigne un técnico, el campo `Trabajo.tecnicoAsignado` no sea nulo, ni el atributo `Tecnico.trabajoAsignado` tampoco lo sea.

Comprobar si al dar un presupuesto un técnico a un trabajo, el campo Trabajo.presupuesto fuera actualizado.

Comprobar que al dar un parte de técnico, al acceder a `Trabajo.parteTecnico`, éste no sea nulo y al obtener sus atributos, estas contengan los valores que hayamos definido previamente.

Comprobar que al dar un parte de trabajo, al acceder a `Trabajo.parteTrabajo`, este no sea nulo y al obtener sus atributos, estas contengan los valores que hayamos definido previamente.

De componentes

Debido a que Ruby On Rails soporta la creación de APIs, combinaremos algunas cosas de comprobación de la página web y de la API:

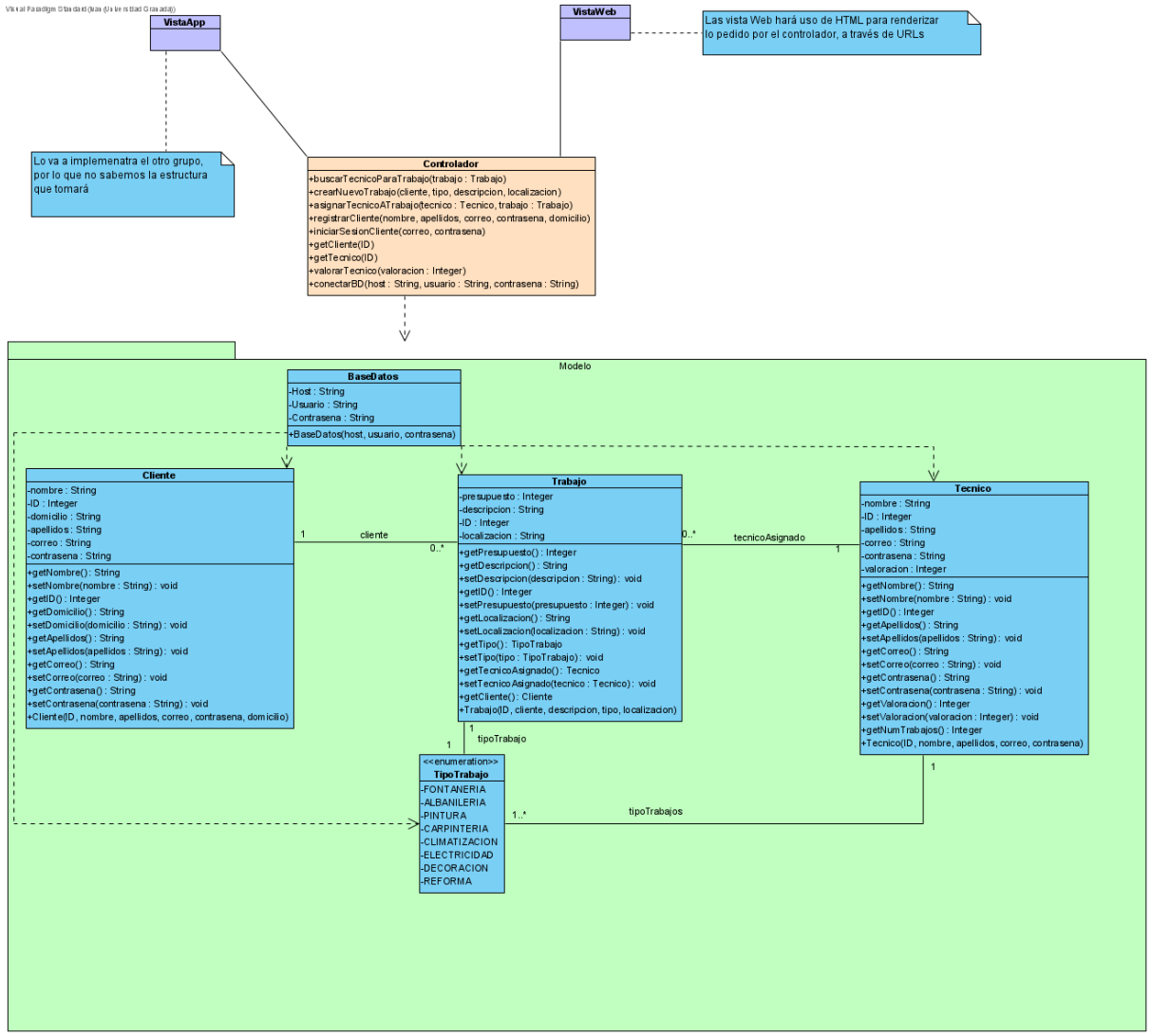
- Si se rellena todos los campos del formulario de Cliente, debería poder crearse el Cliente adecuadamente. Lo mismo aplicaría con solo los obligatorios.
- Si se rellena con algún valor no numérico donde no corresponde, o no se rellenan algún campo obligatorio, debería poder avisarse y no crear el Cliente. De manera alternativa, se podría dejar bloqueado el botón de creación
- Si en la API se llama a un Cliente o Trabajo con ID superior al de número de trabajos o de un trabajo borrado, debería responder con un error 404 Not Found
- Si se hace una actualización a través de API, si se omite un campo sería interesante que la Web o la Aplicación pudieran poner los campos que faltasen en caso de que la propia API no manejara tal salvaguarda
- A la hora de crear un Trabajo, si no hay un Técnico asignado todavía, se debería notificar en una alerta o ventana. De haberlo, se mostraría el nombre y contacto del técnico que ha aceptado.

De integración

- Vamos a intentar comprobar el flujo de interacción básica de los clientes. Para ello:
    - Registramos un Cliente. Dentro de ese registro:
        - No pondremos un campo obligatorio, lo que implicaría que no se haga el registro.
        - Pondremos todos los puntos obligatorios, creando así el registro.
    - Iniciamos sesión como Cliente:
        - Con un usuario incorrecto, dando fallo
        - Con un usuario correcto y contraseña incorrecta, dando fallo
        - Con un usuario y contraseña correctas, redirigiendo la página
    - Creamos un Trabajo:
        - Con el campo del presupuesto relleno con un valor no numérico, lo que daría fallo
        - Con algún campo obligatorio sin rellenar, lo que daría fallo.
        - Con todos los campos rellenos, lo que redirigiría a la página que muestra el Trabajo.
    - Si se hace que el Cliente active la búsqueda, al pulsar el botón se empiece a realizar peticiones por la aplicación.
    - Si se hace automáticamente, las peticiones se realizarán nada más se haya creado.

Diagrama de clases de análisis

A raíz de todo lo expuesto anteriormente, se ha decidido crear el siguiente diagrama de clases.



Si bien se usaría una BD para ello, algunas de esas clases se convertirán en tablas de la Base de Datos. Generalmente eso sería el modelo, quitando la clase BaseDatos para su conexión.