Índice

- 1. Introducción
- 2. Objetivos
- 3. Tecnologías escogidas y justificación
  - 3.1. Tecnologías escogidas y justificación
  - 3.2. Motor de bases de datos
  - 3.3. Frameworks seleccionados
- 4. Diseño de la aplicación
  - 4.1. Casos de uso
  - 4.2. Modelo de dominio

1. Introducción

Este proyecto consiste en el desarrollo de una aplicación web capaz de realizar un itinerario inteligente para un técnico comercial conforme a criterios preestablecidos.
El usuario podrá acceder a "RutCom" desde cualquier dispositivo inteligente. Su diseño sencillo e intuitivo esconde una potente herramienta que integra todo lo necesario para el día a día de sus empleados en movilidad.

2. Objetivos

La razón de la elaboración de este proyecto viene dada por la necesidad que tienen las empresas en cuyo portfolio incluya en algún servicio de movilidad tanto de reparto como de carácter comercial. Con esta aplicación conseguimos tanto un abaratamiento en los costes de desplazamiento como una mejora en el planning estrategico de cualquier empresa que requiera movilidad.

3. Tecnologías escogidas y justificación

3.1. Tecnologías escogidas y justificación

La aplicación se desarrolla en su mayor parte usando lenguaje de servidor PHP. No obstante, existen partes de la misma realizadas en lenguaje cliente Javascript y su biblioteca multiplataforma `Jquery`.
Cabe mencionar que todo el software utilizado es libre, con lo que la empresa tendrá un considerable ahorro en concepto de licencias. El software seleccionado no lo ha sido sólo por gratuito, sino porque además es unas de las tecnologías más utilizada en la actualidad en el desarrollo de aplicaciones web debido a su fiabilidad y a su versatilidad.

Pero lo más importante a destacar es que el lenguaje de programación escogido va asociado al framework. La selección del framework se desarrolla en el punto 3.3

3.2. Motor de bases de datos

El motor de bases de datos usado es `MySql`. Las razones de su elección son las siguientes:

- Su adquisición es gratuita, lo que permite reducción de costes para el cliente.
- Es multiplataforma para Windows, Linux y Mac (los sistemas operativos más extendidos) con lo cual se podrá disponer él en cualquiera de estos.
- Es un motor muy extendido en la comunidad de desarrolladores, con lo que conseguir ayuda es muy sencillo.
- La labor de mantenimiento de una base de datos MySql es muy fácil debido a que presenta menos funciones frente a otros sistemas gestores. Esto, aunque pueda parecer una desventaja, tiene a su favor que el mantenimiento de la aplicación lo puede llevar el propio desarrollador, sin tener que recurrir a un administrador de bases de datos.
- Es escalable, lo cual nos da una ventaja con vistas al futuro.

3.3. Frameworks seleccionados

3.3.1 Frontend

Gmaps.js

GMaps.js es una librería JavaScript que se basa en Google Maps y que permite publicar mapas en la web de forma extremadamente sencilla. Utiliza **jquery** lo que permite reducir el código al máximo y hacerlo fácilmente entendible. Nos permite crear mapas con marcadores, rutas, geolocalización, perfiles longitudinales… y muchas otras funciones.

FullCalendar

Fullcalendar es un plugin de jQuery escrito por Adam Shaw que presenta datos en forma de calendario o agenda. Puede cargar los eventos directo por AJAX (aunque también por otros mecanismos).

Alpine.js

Alpine.js es un pequeño marco de JavaScript declarativo que le permite crear componentes interactivos simples en la página. Perfectamente emparejado con Livewire y por el mismo creador.

Tailwind CSS

Con las clases de utilidad de Tailwind, está escribiendo CSS personalizado sin CSS. Cree sus propios diseños personalizados con la facilidad de Bootstrap y la flexibilidad de CSS escrito a mano.

3.3.2 Backend

El framework para la parte del backend usado es `Laravel 8`. Las razones de su elección son las siguiente:

- Es un framework gratuito.
- Lleva muy bien a la práctica el concepto Modelo-Vista-Controlador, lo cual es muy recomendable para el mantenimiento de aplicaciones.
- Está construido en lenguaje PHP, lo cual nos da una ventaja en el sentido de que este lenguaje es de los más extendidos en el mundo actualmente.
- La comunidad de desarrolladores Laravel es muy amplia. Como consecuencia de esto encontraremos infinidad de módulos reutilizables ya probados que podemos añadir a nuestro código de forma sencilla y que nos ahorrará es tener que crearlos y probarlos nosotros mismos.
- La ayuda de Laravel es muy buena e intuitiva.
- Los módulos existentes están muy estandarizados desde el punto de vista del código, con lo que estamos facilitando la interoperatividad.
- Buena comunicación entre el back-end y el front-end.
- Organiza las funcionalidades en paquetes o Bundles. Estos, si no vienen con el core pueden instalarse y podremos aprovechar su funcionalidad.
- El core de Laravel se basa en la integración de cuatro productos muy potentes ya existentes en el mercado:

- `Blade:` Es el gestor de plantillas html. Resuelve muy bien la separación entre las mismas y los controladores. Permite herencia y bloques html, lo que permite la no repetición de código innecesario.
- `Eloquent:` Es un ORM (Object Relational Mapping) consistente en la equivalencia entre tablas de la base de datos y entidades. Esto permite al desarrollador olvidarse del lenguaje SQL y centrarse en su aplicación.
- `Swiftmailer:` Es una librería para el envío de correos electrónicos. Permite usar varios transporters como SMTP o GMAIL.
- `Livewire:` Componentes de vista de Laravel, entregados sin problemas a sus usuarios a través de JavaScript.

4. Diseño de la aplicación

4.1. Funcionalidades de la aplicación

4.1.1 Vistas de la aplicación

- Clientes
  - Importará datos de clientes de los principales formatos: Hojas excel, vCard, CSV
  - Geolocalización de clientes a partir de dirección postal
  - Herramientas de etiquetado de clientes
  - Datos calculados personalizados por clientes:
    - Venta
    - Valoraciones
    - Tiempo
    - Tiempo estimado de visita
    - Datos agregados personalizados
  - Informes de clientes:
    - Una vez visitado el cliente envío de informe de resultado de visita
- Rutas:
  - Creación de rutas optimizadas en función de clientes a visitar
  - Rutas calculadas por tiempo de visita o desplazamiento
  - Rutas calculadas por datos personalizados o etiquetas
  - Estimación de costes por ruta
  - Dashboard de kilometraje
  - Información de incidencias durante la jornada
- Agenda:
  - Creación de plan de visitas semanal, mensual o anual.
  - Integración con calendarios comerciales Google Calendar o Calendario Apple
  - Notificaciones automáticas a clientes por atrasos en ruta.
- Fichaje
  - Control de fichaje en cumplimiento de la legislación vigente R.D. 8/2019

4.1.2 Sistema de Login

Jetstream

Laravel Liveware simplemente nos ofrece un scafolding o esqueleto para poder partir de algo más que un proyecto.

Laravel LIveware nos ofrece un esquema o funcionalidades sencillas ya implementadas; una vista de perfil, modificar datos vacíos, cargar un avatar, cambiar la contraseña, crear Tokens de autenticación con Laravel Sanctum y manejo de equipos que pueden funcionar en base a un sistema de roles; todas estas características pueden ser apagadas o removidas fácilmente de nuestro proyecto.

Laravel Livewire nos permite un completo sistema de doble autenticación, para esto podemos emplear aplicaciones como Google o Microsoft Authenticator, 1Password, etc; simplemente habilitamos la opción en el módulo correspondiente y seguimos el resto de los pasos.

Luego, si cerramos sesión vas a necesitar ingresar el código que te genera tu app de autenticación además de la pareja de usuario y contraseña.

4.2. Modelo de datos

Entidades

Usuario

- Se crea como entidad porque guardara la información de todos los usuarios de la aplicación
- Atributos: ID, usuario, contraseña.
- Relaciones: se relaciona con perfil y persona, y es débil de persona.

Perfil

- Justificación: Hemos creado esta entidad porque agrupamos los permisos por perfil, este perfil tiene que ver con los permisos que va a tener el usuario en la aplicación.
- Atributos: ID, nombre_perfil.
- Relaciones: Se relaciona con permisos y usuario.

Permiso

- Tiene la suficiente independencia como para que sea una entidad por sí misma, guardara el nombre de una función de nuestra aplicación con su acción.
- Atributos: ID, Nombre_entidad, FK.
- Se relaciona con perfil y acción.

Acción

- Hemos creado esta entidad porque la información que va a guardar esta se va a repetir muchas veces.
- ID, Nombre_acción.
- Se relaciona con permiso, y es débil de esta.

Persona

- Necesitamos registrar los datos personales del trabajador, es el núcleo de nuestra base de datos, si se borra una persona habría que borrar todo lo relacionado con ella.
- Atributos: ID, nombre, apellidos, DNI/NIE, NSS, foto y dirección.
- Se relaciona con usuario, informe, comunicación, recordatorio, horario, evento, ubicación, categoría profesional, ruta, control horario, correo electrónico y teléfono.

Categoría persona

- Es entidad dado que sus valores se van a repetir. Se refiere a la categoría profesional.
- Id, Nombre_categoría.
- Se relaciona con "persona".

Teléfono

- Se crea como una entidad porque se guardarán varios teléfonos de una misma persona o cliente.
- Atributos: Teléfono
- Se relaciona con persona y cliente, es débil de estas.

Email

- Se crea como una entidad porque se guardarán varios emails de una misma persona.
- Email
- Se relaciona con persona y es débil de esta

Horario

- Justificación: Se crea como entidad porque deseamos guardar el horario de todas las personas según se firma en el contrato, es un dato que se va a repetir.
- Atributos: Id, Nombre horario, FK_Turno
- Relaciones: Persona, turno.

Turno

- Justificación: Se crea como entidad porque un horario puede estar formado por uno o más turnos, ahí se guardará la información de cada periodo de trabajo.
- Atributos: ID,
- Relación con horario.

Control Horario

- Justificación: Se crea como entidad porque contendrá dos entidades y gracias a él se activará la ubicación real.
- Atributos: FK_persona. Fecha/hora de entrada, fecha/hora de salida.
- elaciones: Con "Persona" y es débil de ésta ya que si no existe la persona no existirá el control horario de esa persona.

Evento

- Justificación: Se crea como entidad porque guardará cualquier cita que tenga la persona.
- Atributos: Id, Nombre_evento, fecha_evento, hora_evento, anotaciones y FKeys.
- Relaciones: Persona, cliente, informe, ubicación fija y recordatorio

Recordatorio

- Justificación: Es un aviso en el calendario y porque es una función que avisa de cierto evento.
- Atributos: Id, texto_recordatorio, dia_recordatorio, fecha_recordatorio
- Relaciones: Persona, evento, y comunicación.

Comunicación

- Justificación: Se crea como entidad porque se desean guardar las comunicaciones que se tienen con los clientes.
- Atributos: Id, fecha y hora, contenido, medio_comunicación.
- Relaciones: Persona, cliente y recordatorio.

Medio Comunicación

- Justificación: Se crea como entidad porque se va a repetir los medios.
- Atributos Id, nombre.
- Relaciones: Comunicación.

Cliente

- Justificación: Se crea como entidad porque deseamos guardar los datos de todos los clientes.
- Atributos: Id, Nombre, Apellido, DNI/CIF,
- Relaciones: Evento, Comunicación, Teléfono, e-mail y Ubicación fija

Categoría cliente

- Justificación: Se crea como entidad porque queremos diferenciar entre los distintos tipos de cliente.
- Atributos: Id, Nombre_categoría.
- Relaciones: Cliente.

Ubicación

- Justificación: Es la base de la aplicación así que necesitamos guardar dichos datos, dicha entidad nos va a permitir guardar la planificación de las rutas, será una generalización de los puntos cambiantes y de los puntos fijos.
- Atributos: Latitud, longitud, nombre_ubicación, detalle_ubicación.
- Relaciones:
- Se relaciona con persona y ruta

Cambiante

- Es una entidad porque es una especialización de ubicación. Se refiere a la ubicación en tiempo real. Empieza a registrar valores en el momento de check-in.
- Los atributos serán los mismo que ubicación y añadiremos la fecha y hora.
- Relación con ruta.

Fija

- Justificación: Es una entidad porque es una especialización de ubicación. Se refiere a puntos fijos como son los eventos, los domicilios, etc...
- Atributos: Latitud, longitud, nombre_ubicación,detalle_ubicación.
- Relaciones:se relaciona con el tag, el cliente y evento.

Tag

- Justificación: Es una entidad porque se refiere al tipo de ubicación (Empresa, domicilio, etc..) y se va a repetir.
- Atributos: Id y nombre.
- Relaciones: Se relaciona con ubicación fija.

Ruta

- Hemos creado esta entidad porque deseamos guardar un conjunto de puntos fijos que se van a realizar en un periodo de tiempo y debemos saber quién la crea y quien la recorre.
- ID, nombre y Fks.
- Se relaciona doblemente con persona, ubicación y cambiante.

Informe

- Justificación: Será información que se guardará después de cada evento y se refiere a todas las entidades relacionadas con el mismo.

- Atributos: Id, comentario.
- Relaciones: Persona y evento.

Relaciones

De cada relación indica lo siguiente:

- Nombre de la relación
- Justificación
- Tablas implicadas
- Cardinalidad de la relación
- Posibles dependencias: Por ejemplo: Cuándo se borra un registro de una de las tablas ¿Qué sucede con el registro relacionado? ¿Se borra también?

Persona/Crea Ruta

- Cardinalidad de la relación: 1...N
- Posibles dependencias: No existe dependencia de borrado. Tampoco existe dependencia de identidad ni de existencia.

Persona/Recorre Ruta

- Cardinalidad de la relación: 1...N
- Posibles dependencias: No existe dependencia de borrado. Tampoco existe dependencia de identidad ni de existencia.

Persona/Categoría Profesional

- Cardinalidad de la relación: 1...1
- Posibles dependencias: No existe dependencia de borrado. Tampoco existe dependencia de identidad ni de existencia.

Persona/Ubicación

- Cardinalidad de la relación: 1...N
- Posibles dependencias: Existe una dependencia de existencia de la ubicación sobre la persona, es decir, al borrar a una persona, sus ubicaciones deben ser borradas en cascada.

Persona/Evento

- Cardinalidad de la relación: N...M
- Posibles dependencias: No existe dependencia de borrado. Tampoco existe dependencia de identidad ni de existencia.

Persona/Horario

- Cardinalidad de la relación: 1...1
- Posibles dependencias: No existe dependencia de borrado. Tampoco existe dependencia de identidad ni de existencia.

Persona/Recordatorio

- Cardinalidad de la relación: 1...N
- Posibles dependencias: Existe una dependencia de existencia del recordatorio sobre la persona, es decir, si se elimina a la persona, sus recordatorios se borrarán en cascada.

Persona/Comunicación

- Cardinalidad de la relación: N...M
- Posibles dependencias: No existe dependencia de borrado. Tampoco existe dependencia de identidad ni de existencia.

Persona/Informe

- Cardinalidad de la relación: 1...N
- Posibles dependencias: No existe dependencia de borrado. Tampoco existe dependencia de identidad ni de existencia.

Persona/Teléfono

- Cardinalidad de la relación: 1...N

- Posibles dependencias: Existe una dependencia de existencia del teléfono sobre la persona, es decir, si eliminamos a la persona, se deben eliminar sus teléfonos en cascada.

Persona/E-mail

- Cardinalidad de la relación: 1...N
- Posibles dependencias: Existe una dependencia de existencia del email sobre la persona, es decir, si eliminamos a la persona, se deben eliminar sus emails en cascada.

Persona/Control Horario

- Cardinalidad de la relación: 1...N
- Posibles dependencias: Existe una dependencia de existencia del control horario sobre la persona, es decir, si el eliminamos a la persona, se deben eliminar su control horario en cascada.

Persona/Usuario

- Cardinalidad de la relación: 1...1
- Posibles dependencias: Existe una dependencia de existencia, al borrar la persona debe eliminarse el usuario.

Usuario/Perfil

- Cardinalidad de la relación: 1…N
- Posibles dependencias: No existen dependencias entre ambas entidades.

Perfil/Permiso

- Cardinalidad de la relación: 1...N
- Posibles dependencias: No existen dependencias entre ambas entidades.

Permiso/Acción

- Cardinalidad de la relación: 1...N
- Posibles dependencias: No existen dependencias entre ambas entidades.

Ruta/Ubicación

- Cardinalidad de la relación: N...M
- Posibles dependencias: No existen dependencias entre ambas entidades.

Ruta/Cambiante

- Cardinalidad de la relación: 1...N
- Posibles dependencias: Existe una dependencia de existencia, si borramos una ruta, hay que borrar los cambiantes.

Fija/Tag

- Cardinalidad de la relación: N...M
- Posibles dependencias: No existen dependencias entre ambas entidades.

Fija/Cliente

- Cardinalidad de la relación: 1...N
- Posibles dependencias: Existe una dependencia de existencia, si borramos un cliente tenemos que borrar sus puntos fijos.

Fija/Evento

- Cardinalidad de la relación: 1...1
- Posibles dependencias: No existen dependencias entre ambas entidades.

Evento/Informe

- Cardinalidad de la relación: 1...1
- Posibles dependencias: No existen dependencias entre ambas entidades.

Evento/Cliente

- Cardinalidad de la relación: N...M
- Posibles dependencias: No existen dependencias entre ambas entidades.

Evento/Recordatorio

- Cardinalidad de la relación: 0...N

- Posibles dependencias: Existe una dependencia de existencia, si eliminamos un evento, tenemos que borrar ese recordatorio.

Horario/turno

- Cardinalidad de la relación: N...M
- Posibles dependencias: No existen dependencias entre ambas entidades.

Recordatorio/Comunicación

- Cardinalidad de la relación: 0...N
- Posibles dependencias: Existe una dependencia de existencia, si borramos la comunicación, hay que borrar el recordatorio.

Comunicación/Medio

- Cardinalidad de la relación: 1...N
- Posibles dependencias: No existen dependencias entre ambas entidades.

Cliente/Categoría

- Cardinalidad de la relación: 1...N
- Posibles dependencias: No existen dependencias entre ambas entidades.

Cliente/Email

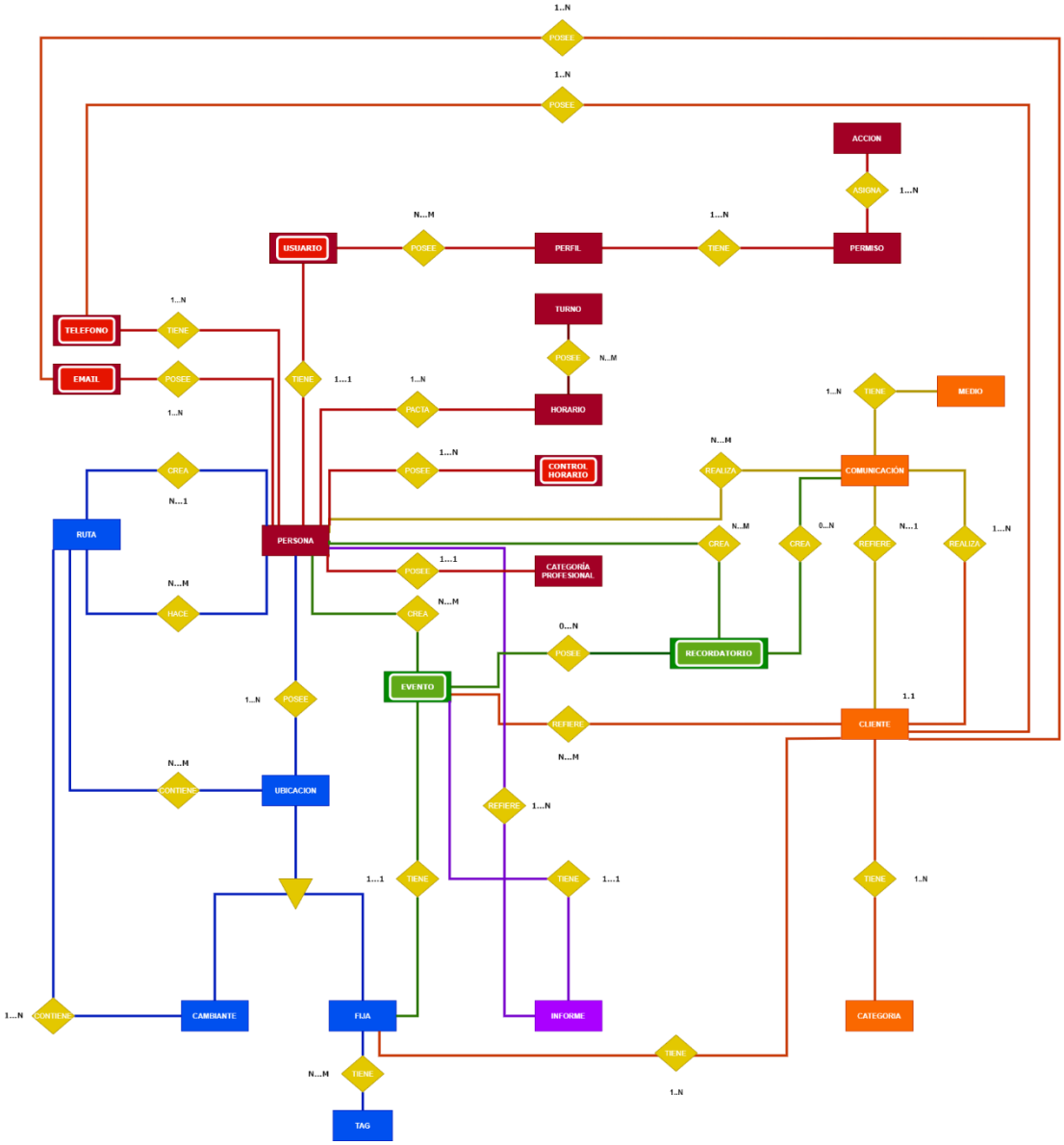
- Cardinalidad de la relación: 1...N
- Posibles dependencias: Existe una dependencia de existencia, si borramos un cliente, hay que borrar también el email.

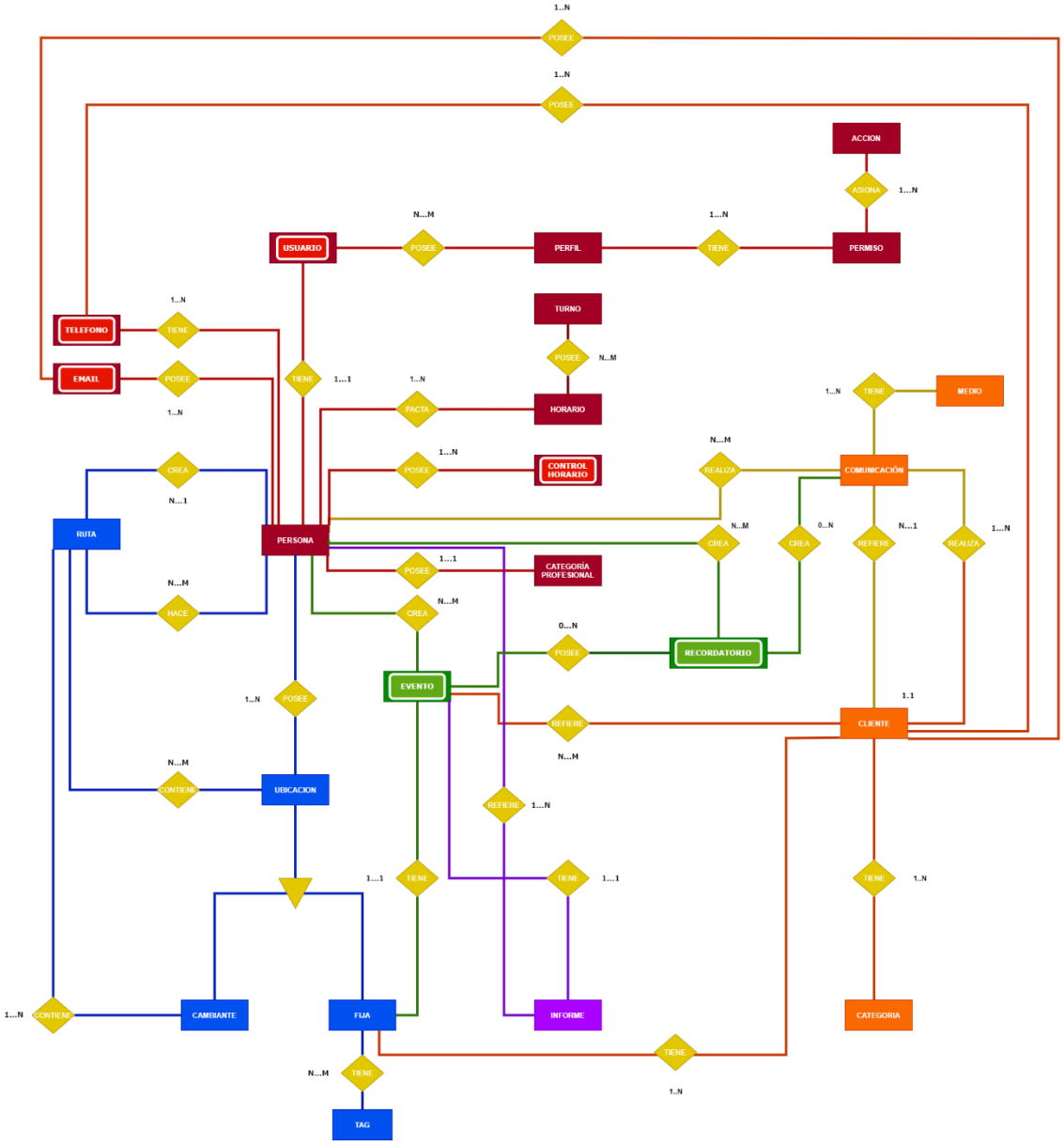
Cliente/Teléfono

- Cardinalidad de la relación: 1...N
- Posibles dependencias: Existe una dependencia de existencia, si borramos un cliente, hay que borrar también el teléfono.

Cliente/Comunicación

- Cardinalidad de la relación: 1...N
- Posibles dependencias: No existen dependencias entre ambas entidades.


1..N
POSEE
1..N
POSEE
ACCION
ASIGNA
1...N
N...M
USUARIO
POSEE
PERFIL
1...N
TIENE
PERMISO
TURNO
1...N
TELEFONO
TIENE
POSEE
N..M
EMAIL
POSEE
TIENE
1...1
1...N
PACTA
HORARIO
1...N
MEDIO
TIENE
1...N
N...M
CREA
POSEE
1...N
CONTROL HORARIO
REALIZA
COMUNICACIÓN
N...1
RUTA
PERSONA
CREA
N...M
CREA
0...N
REFIERE
N...1
REALIZA
1...N
POSEE
1...1
CATEGORÍA PROFESIONAL
N...M
HACE
CREA
N...M
0...N
POSEE
RECORDATORIO
EVENTO
1...N
POSEE
1.1
REFIERE
CLIENTE
N...M
N...M
CONTIENE
UBICACION
REFIERE
1...N
1...1
TIENE
TIENE
1...1
TIENE
1..N
1...N
CONTIENE
CAMBIANTE
FIJA
INFORME
CATEGORIA
TIENE
N...M
TIENE
1..N
TAG


1...N
POSEE
1...N
POSEE
ACCION
ASIGNA
1...N
N...M
USUARIO
POSEE
PERFIL
1...N
TIENE
PERMISO
TURNO
1...N
TELEFONO
TIENE
POSEE
N...M
EMAIL
POSEE
TIENE
1...1
1...N
PACTA
HORARIO
1...N
1...N
TIENE
MEDIO
N...M
CREA
POSEE
1...N
CONTROL HORARIO
REALIZA
COMUNICACIÓN
N...1
RUTA
PERSONA
N...M
CREA
0...N
CREA
N...1
REFIERE
1...N
REALIZA
1...1
POSEE
CATEGORÍA PROFESIONAL
N...M
HACE
CREA
N...M
0...N
POSEE
RECORDATORIO
EVENTO
1...N
POSEE
1.1
REFIERE
CLIENTE
N...M
N...M
CONTIENE
UBICACION
REFIERE
1...N
1...1
TIENE
1...1
TIENE
TIENE
1..N
1...N
CONTIENE
CAMBIANTE
FIJA
INFORME
CATEGORIA
TIENE
N...M
TIENE
1..N
TAG