



**Universidad Autónoma de San  
Luis Potosí  
Facultad de Ingeniería**



**Área de Ciencias de la Computación**

**Equipo:**

Arreguín Bustos José Emmanuel

Briones Medina Danna Itzel

Martínez Sustaita Keivin Damagd

Mondragón Cuevas Luis Felipe

Sánchez Angulo Andrés de Jesús

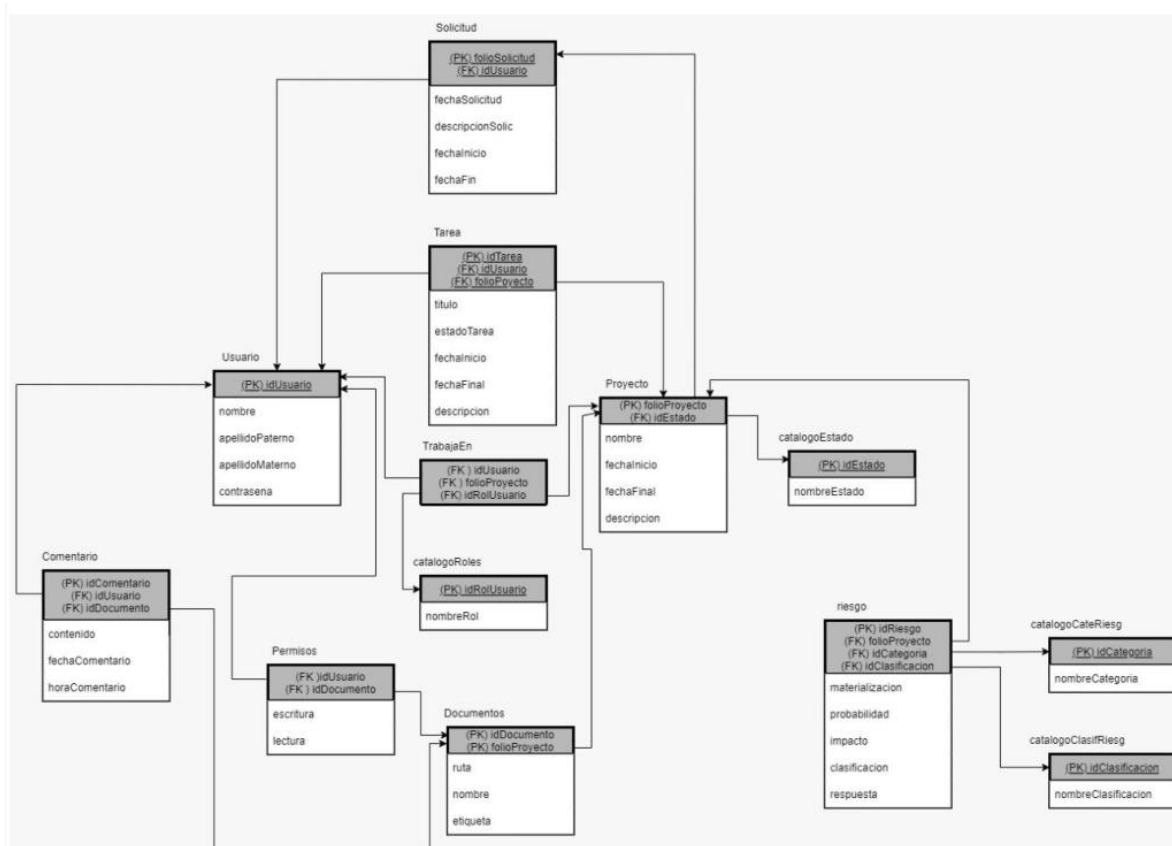
**Maestro:** Francisco Javier Torres Reyes

**Materia:** Proyectos Computacionales II

***Registro de Actividades del Departamento de  
Informática***

Semestre: 2021 – 2022/II

De acuerdo con el modelo relacional planteado en las materias de PCI y administración de proyectos II, nuestra base de datos contendría los siguientes atributos:



Basándonos en lo planteado, las tablas de este modelo cambiaron un poco, esto debido al trabajo de desarrollo que hubo además de inconvenientes con atributos que no se habían tomado en cuenta. A continuación, listare como quedaron actualmente las migraciones que se encuentran en el proyecto de laravel.

## Usuario

```

Schema::create('users', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->string('email')->unique();
    $table->string('apellidoPaterno');//*
    $table->string('apellidoMaterno');//*
    $table->string('tipoUsuario')->nullable();//*
    $table->timestamp('email_verified_at')->nullable();
    $table->string('password');
    $table->rememberToken();
    $table->timestamps();
});
    
```

En la tabla de usuario no hubo ningún cambio significativo más que nada cambios de nombre, y los añadidos son parte del framework de laravel debido a que integramos los usuarios que maneja laravel por defecto.

## Catalogo de Estados

```
Schema::create('catalogo_estados', function (Blueprint $table) {  
    $table->id();  
    $table->string('nombreEstado');  
    $table->timestamps();  
});
```

La tabla de catálogo de estados no contiene ningún cambio.

## Proyectos

```
Schema::create('proyectos', function (Blueprint $table) {  
    $table->id();  
    $table->unsignedBigInteger('idEstado');  
    $table->foreign('idEstado')->references('id')->on('catalogo_estados');  
    $table->string('nombre');  
    $table->string('fechaInicio');  
    $table->string('fechaFinal');  
    $table->text('descripcion');  
    $table->timestamps();  
});
```

La tabla de proyectos tampoco contiene cambios.

## Solicitudes

```
Schema::create('solicitudes', function (Blueprint $table) {  
    $table->id();  
    $table->string('asuntoSolicitud');  
    $table->unsignedBigInteger('idUserarioC');//idUserarioC llave foranea de u  
    $table->unsignedBigInteger('idUserarioD');//idUserarioD llave foranea de u  
    $table->unsignedBigInteger('idEstado');//Estado en el que se encuentra l  
    $table->foreign('idUserarioC')->references('id')->on('users');  
    $table->foreign('idUserarioD')->references('id')->on('users');  
    $table->foreign('idEstado')->references('id')->on('catalogo_estados');  
    $table->string('fechaSolicitud');  
    $table->text('descripcionSolic');  
    $table->string('documentoDescripcion');  
    $table->string('fechaInicio');  
    $table->string('fechaFin');  
    $table->text('retroalimentacion');  
    $table->string('idDocumento');  
    $table->timestamps();  
});
```

La tabla de solicitudes contiene varios cambios, se le agrega el campo asunto solicitud es donde va el nombre que se desee poner al proyecto, idUsuarioC contiene el id del usuario que crea la solicitud en este caso el cliente, idUsuarioD es el usuario al que se le asignara el rol de líder del proyecto. El campo idEstado hace referencia al estado en el que se encuentra nuestra solicitud, un ejemplo de estados es Enviada la cual implica que la solicitud fue procesada y enviada al administrador para su posterior aprobación. Para que el administrador pueda hacer comentarios sobre la solicitud del proyecto se hizo “retroalimentación” cuya función es hacer llegar los comentarios que tiene el administrador acerca de la solicitud. También se puede poner una descripción en un documento de Word o algún archivo de texto ya que este se almacena en “idDocumento” para su lectura y almacenamiento.

#### Catalogo Categoría Riesgos

```
Schema::create('catalogo_cate_riesgs', function (Blueprint $table) {  
    $table->increments('idCategoria');  
    $table->string('nombreCategoria');  
    $table->timestamps();  
});
```

Sin cambios.

#### Catalogo Clasificación Riesgos

```
Schema::create('catalogo_clasif_riesgs', function (Blueprint $table) {  
    $table->increments('idClasificacion');  
    $table->string('nombreClasificacion');  
    $table->timestamps();  
});
```

Sin cambios.

#### Riesgos

```
Schema::create('riesgos', function (Blueprint $table) {  
    $table->id();  
    $table->timestamps();  
});
```

Aun no se ha implementado la tabla de riesgos por lo que dejamos en pausa esta parte del proyecto.

## Tareas

```
Schema::create('tareas', function (Blueprint $table) {
    $table->id();
    $table->string('titulo');
    $table->unsignedBigInteger('idUsuario');//idUsuarioC llave foranea de usua
    $table->unsignedBigInteger('folioProyecto');
    $table->unsignedBigInteger('estadoTarea');//Estado en el que se encuentra
    $table->unsignedBigInteger('faseTarea');//Estado en el que se encuentra la
    $table->foreign('idUsuario')->references('id')->on('users');
    $table->foreign('folioProyecto')->references('id')->on('proyectos');
    $table->foreign('estadoTarea')->references('id')->on('catalogo_estados');
    $table->foreign('faseTarea')->references('id')->on('catalogo_estados');
    $table->string('fechaInicio');
    $table->string('fechaFinal');
    $table->text('descripcion');
    $table->timestamps();
});
```

FolioProyecto nos permite saber en qué proyectos están asignadas las tareas, el estadoTarea es para ver en que estado se encuentra la tarea ej. Aprobada, rechazada, etc. Las fases de las tareas están definidas por el modelo Mic-Agile del cliente las cuales son Analizar, Diseñar, Desarrollar, Evaluar.

## Catalogo Roles

```
Schema::create('catalogo_roles', function (Blueprint $table) {
    $table->id();
    $table->string('nombreRol');
    $table->timestamps();
});
```

Sin cambios.

Trabaja en

```
Schema::create('trabaja_ens', function (Blueprint $table) {
    $table->id();
    $table->unsignedBigInteger('idUsuario');
    $table->unsignedBigInteger('folioProyecto');
    $table->unsignedBigInteger('idRolUsuario');
    $table->foreign('idUsuario')->references('id')->on('users');
    $table->foreign('folioProyecto')->references('id')->on('proyectos');
    $table->foreign('idRolUsuario')->references('id')->on('catalogo_roles');
    $table->timestamps();
});
```

Sin cambios.

## Documentos

```
Schema::create('documentos', function (Blueprint $table) {
    $table->id();
    $table->string('ruta');
    $table->string('nombre');
    $table->string('etiqueta');
    $table->unsignedBigInteger('folioProyecto');
    $table->foreign('folioProyecto')->references('id')->on('proyectos');
    $table->timestamps();
});
```

El único cambio aquí es que se añadió el campo de folioProyecto para referenciar el proyecto donde se subirán los documentos que sean necesarios.

## Permisos

```
Schema::create('permisos', function (Blueprint $table) {
    $table->id();
    $table->timestamps();
});
```

Aun no se implementan los permisos a los usuarios.

## Comentarios

```
Schema::create('comentarios', function (Blueprint $table) {
    $table->id();
    $table->unsignedBigInteger("idUserario");
    $table->foreign('idUserario')->references('id')->on('users');
    $table->unsignedBigInteger("idDocumento");
    $table->foreign('idDocumento')->references('id')->on('documentos');
    $table->string('contenido');
    $table->timestamps();
});
```

Sin cambios. Aun no implementado.

## Respaldo de los documentos

```
Schema::create('documentos__backups', function (Blueprint $table) {
    $table->id();
    $table->string("ruta");
    $table->unsignedBigInteger("folioDocumento");
    $table->foreign('folioDocumento')->references('id')->on('documentos');
    $table->timestamps();
});
```

Esta tabla es nueva y se hizo para facilitar el guardado de los documentos ya almacenados en el sistema, ya que se requiere que al menos se guarde una copia del documento antes de sobrescribirlo.

Para probar el sistema se hizo uso de seeders, para dejar algunos usuarios de distintos tipos y realizar pruebas de creación de solicitudes, proyectos, tareas y subida de documentos. Esperamos dejar los usuarios al principio de la implementación y después removerlos para que no sea un riesgo de seguridad.

Los modelos nos permiten realizar acciones a la base de datos y especificar los datos que vamos a requerir dependiendo del modelo, el modelo user sea crea al implementar las sesiones en laravel y solo cambiamos algunos campos por los que necesitábamos.

### Modelo Trabaja En

```
class TrabajaEn extends Model
{
    protected $table="trabaja_ens";

    public function Usuario(){
        return $this->belongsTo(User::class,'idUserario');
    }

    public function Proyecto(){
        return $this->belongsTo(Proyecto::class,'folioProyecto');
    }

    public function Rol(){
        return $this->belongsTo(catalogoRoles::class,'idRolUsuario');
    }
}
```

Utilizamos el belongsTo para hacer la relación de las llaves foráneas idUsuario de la tabla usuario y folioProyecto a su respectiva tabla.

### Tarea

```
class Tarea extends Model
{
    //use HasFactory;
    protected $table="tareas";

    public function Proyecto(){
        return $this->belongsTo(Proyecto::class,'folioProyecto');
    }

    public function Usuario(){
        return $this->belongsTo(User::class,'idUserario');
    }

    public function Estado(){
        return $this->belongsTo(catalogoEstado::class,'estadoTarea');
    }

    public function Fase(){
        return $this->belongsTo(catalogoEstado::class,'faseTarea');
    }
}
```

Se usa belongsTo para hacer la relación de la llave foránea con la llave principal de la tabla a la que pertenece respectivamente.

#### Proyecto

```
class Proyecto extends Model
{
    protected $table="proyectos";

    public function Integrantes(){
        return $this->hasMany(TrabajaEn::class,'folioProyecto','id');
    }

    public function Tareas(){
        return $this->hasMany(Tarea::class,'folioProyecto','id');
    }

    public function Documentos(){
        return $this->hasMany(Documentos::class,'folioProyecto','id');
    }
}
```

El modelo del proyecto utiliza la relación de hasMany ya que un proyecto tiene muchas tareas realizándose o en proceso, varios documentos almacenados e integrantes del equipo trabajando simultáneamente en este.

#### Documentos

```
class Documentos extends Model
{
    protected $table="documentos";

    public function VersionAnterior(){
        return $this->hasOne(Documentos_Backup::class,'folioDocumento');
    }

    public function Comentarios(){
        return $this->hasMany(Comentario::class,'idDocumento','id');
    }
}
```

En el modelo de documentos tenemos dos tipos de relación, la de folioDocumento en la que asociamos la llave foránea a la principal de documentos backup para así poder guardar una copia de la versión anterior de un documento. Y la de comentario en la que asociamos el id del documento con el id del comentario para que un usuario pueda hacer varios comentarios en un solo documento.



Se decidió agrupar las views, en auth, layouts, vista-principal y vista-proyecto, las views de auth se generan automáticamente por laravel solo realizamos cambios estéticos en el login y dejamos una pagina sin estilos para el registro ya que esta parte debe ser automatizada.

### Vista del login



### Vista de registro



La base de la pagina se encuentra en layouts donde incluiremos las diferentes partes de la pagina para no poner todo el código en un solo archivo php, en base.blade.php se agregan las partes de la cabecera y el pie de la pagina web que no cambian y se pueden reutilizar. Así como tambien las dos barras de navegación que utilizaremos para movernos por el sistema.



Como podemos ver en la imagen se encuentran dos navbars, la superior es sobre los proyectos activos que tiene el usuario y la ventana de solicitudes en caso de que quiera enviar una solicitud de proyecto.

Las vistas principales son las que tienen que ver con los proyectos y las solicitudes, dentro de la vista inicio.blade.php tenemos los contenedores donde ira el contenido de cada pestaña para esto utilizamos Javascript con Ajax que nos permite navegar por la pagina sin tener que recargar la pagina cada que se realiza un pequeño cambio.

```

<div class="tab-content" id="nav-tabContent" style="margin:20px;">

    <div class="tab-pane fade show active" id="nav-proyecto" role="tabpanel" aria-labelledby="nav-proyecto">
    </div>

    <div class="tab-pane fade" id="nav-solicitud" role="tabpanel" aria-labelledby="nav-solicitud-tab">
    </div>

    <div class="tab-pane fade" id="nav-adminPanel" role="tabpanel" aria-labelledby="nav-adminPanel-tab">
    </div>
</div>

```

Estos serian los contenedores donde colocaremos los datos que obtengamos de Ajax.

```

var ruta = document.getElementById('ruta');

$(document).ready(function(){
    $(document).on('click','#nav-proyecto-tab',function(){
        ruta.innerText = 'Inicio / Mis Proyecto';
        $.ajax({
            type: "GET",
            url: "/proyectos",
            success:function(data){
                $("#nav-solicitud").empty();
                $("#nav-adminPanel").empty();
                $("#nav-proyecto").html(data);
            }
        });
    });
});

```

El script realizado en Javascript, espera a que exista un evento click en la pestaña de proyecto, entonces usamos Ajax para hacer un get mediante el URL /proyectos y en caso de que el Get sea exitoso se elimina lo que haya en los otros divs y se inserta la información que obtuvimos con la consulta de la base de datos.

El Get se utiliza para la ruta de /proyectos que tenemos en el sistema, entonces Ajax realiza la consulta en nuestra hoja de rutas llamada web.php en donde se encuentre una con el /proyectos y que sea un get.

```

Route::get('/proyecto/{id}',[PaginasController::class,'proyecto']->name('proyecto'));
Route::get('/proyectos',[PaginasController::class,'proyectos']);
Route::get('/solicitudes',[PaginasController::class,'solicitudes']);

```

Aquí la señalamos con una flecha roja, por lo que va al controlador de las paginas y busca una función llamada proyectos.

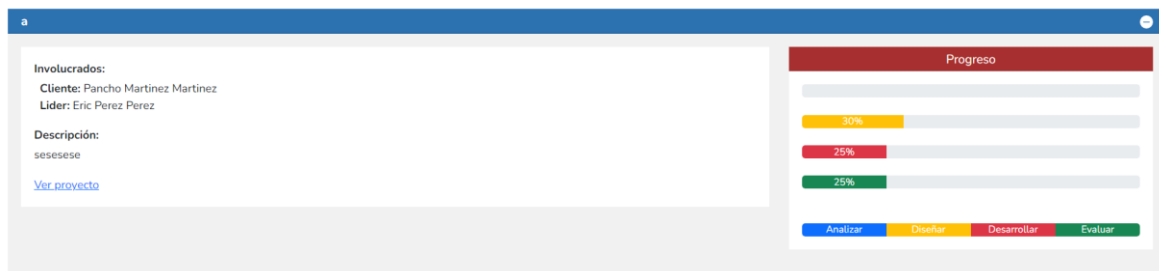
```
public function proyectos(){
    //$proyectos = Proyecto::all();
    $user = Auth::user();

    $relaciones = $user->Relaciones;
    return view('vistas-principal.proyectos',compact('relaciones'));
}
```

En esta función se obtiene el usuario que está haciendo la petición de los proyectos en los que esta involucrado por lo que se obtienen las relaciones de este usuario, esto quiere decir que, si el usuario esta implicado en dos proyectos entonces tiene dos relaciones, pero en cada proyecto puede tener distintos roles. Posteriormente la información obtenida se compacta y se envia a la carpeta vistas-principal al blade proyectos.

```
@foreach($relaciones as $r)
@php
    $p = $r->Proyecto;
@endphp
<div class="card position-relative">
    <div class="card-header">
        <titulo style="font-weight:bold;">{{ $p->nombre}}</titulo>
        <div class="d-flex flex-row-reverse" style="margin-top:-19px;">
            <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi bi-dash-circle">
                <path d="M16 8A8 8 0 1 1 0 8a8 8 0 1 1 16 0zM4.5 7.5a.5.5 0 0 0 1h7a.5.5 0 0 0 0-1h-7z"/>
            </svg>
        </div>
    </div>
    <div class="card-body">
        <div class="row">
            <div class="col-sm-8">
                <div class="card">
                    <div class="card-body">
                        <p class="card-text" style="margin-bottom:3px;font-weight:bold;">Involucrados:</p>
                        @foreach($p->Integrantes as $i)
                            <p class="card-text" style="margin:0px;margin-left:7px;"><rol style="font-weight:bold;">{{ $i->rol }}</rol> {{ $i->nombre }}</p>
                        @endforeach
                    </div>
                </div>
            </div>
            <div class="col-sm-4">
                <div class="card">
                    <div class="card-body">
                        <p style="text-align:center;font-weight:bold;">Progreso</p>
                        <div style="text-align:center;">
                            <div style="width:30%;background-color:yellow;border:1px solid black;margin:2px 0;"></div>
                            30%
                            <div style="width:25%;background-color:red;border:1px solid black;margin:2px 0;"></div>
                            25%
                            <div style="width:25%;background-color:green;border:1px solid black;margin:2px 0;"></div>
                            25%
                        </div>
                        <div style="display:flex;justify-content:center;gap:5px;margin-top:5px;">
                            <div style="width:20px;height:10px;background-color:blue;border:1px solid black;"></div> Analizar
                            <div style="width:20px;height:10px;background-color:yellow;border:1px solid black;"></div> Diseñar
                            <div style="width:20px;height:10px;background-color:red;border:1px solid black;"></div> Desarrollar
                            <div style="width:20px;height:10px;background-color:green;border:1px solid black;"></div> Evaluar
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

Dentro del Blade proyectos obtenemos la variable relaciones y dentro de un foreach obtenemos los proyectos en los que este implicado el usuario, estos proyectos se mostraran de uno por uno, en la card se muestra la información relativa al proyecto como lo es el nombre del proyecto, los integrantes, etc. Se vería así esta sección del proyecto:



La parte de progreso se calcula por la cantidad de tareas de ingeniería que se hayan concluido y por lo tanto se distribuyen estas barras con porcentajes.

La sección de solicitudes se ve así:

The screenshot displays a web interface for managing project requests. At the top, there is a header bar with a blue section on the left and a red section on the right containing a user profile icon and the name 'Eric Perez Perez' with a 'Cerrar Sesión' link. Below the header is a navigation bar with a home icon and the text 'Inicio / Mis Solicitudes'. A secondary bar contains the motto 'Siempre Autónoma. Por mi patria educaré.' and two tabs: 'Mis Proyectos' and 'Solicitud', with the latter being the active tab. The main content area is divided into two panels. The left panel, titled 'a', contains details for a request: 'Nombre del solicitante: Pancho Martinez Martinez', 'Solicitud enviada a: Eric Perez Perez', 'Fecha de solicitud: 2022-05-26', 'Fecha de inicio del proyecto: 2022-05-27', 'Fecha de termino del proyecto: 2022-06-02', and a description 'sesesese'. The right panel, titled 'Estado', shows 'Solicitud aceptada' with a large green checkmark. Below these panels is a button labeled 'Nueva Solicitud'. The footer consists of three columns: 'UASLP' (Universidad Autónoma de San Luis Potosí), 'Facultad de ingeniería', and 'Departamento de informática', each with their respective addresses and contact information.

UASLP	Facultad de ingeniería	Departamento de informática
Universidad Autónoma de San Luis Potosí Álvaro Obregón #64, Col. Centro, C.P. 78000 San Luis Potosí, S.L.P., México Tel. +52 (444) 8262300	Universidad Autónoma de San Luis Potosí Álvaro Obregón #64, Col. Centro, C.P. 78000 San Luis Potosí, S.L.P., México Tel. +52 (444) 8262300	Universidad Autónoma de San Luis Potosí Álvaro Obregón #64, Col. Centro, C.P. 78000 San Luis Potosí, S.L.P., México Tel. +52 (444) 8262300

En este caso de ejemplo se aprobó una solicitud de un proyecto, por eso el estado de la solicitud se encuentra en aprobado. Un usuario puede hacer varias solicitudes de proyectos y estas deben ser aprobadas por el administrador.

Se utiliza un modal para realizar la solicitud, dentro de este se deben llenar todos los campos para poder hacer correctamente el envío de la solicitud, en caso contrario no se enviará la solicitud. Una vez llenado el formulario, se utiliza la siguiente función Javascript para enviar el formulario al controlador correspondiente:

```

$("#modal-formulario").submit(function(e) {
    e.preventDefault(); // avoid to execute the actual
    var formData = new FormData(this);
    $.ajax({
        type: 'POST',
        url: "{{ url('/crearSolicitud')}}",
        data: formData,
        cache: false,
        contentType: false,
        processData: false,
        success: (data) => {
            $('#solicitudProyectoModal').modal('hide');
            $("#nav-solicitud").html(data);
        }
    });
});

```

Una vez que se de clic en el botón de enviar dentro del formulario, se evita que se envíe la solicitud, ya que en este caso usamos Ajax para hacer las peticiones, creamos una nueva variable de tipo FormData que enviamos al url de crearSolicitud en nuestra hoja de rutas.

```

Route::get('/home', [App\Http\Controllers\HomeController::class, 'index'])->name('home');
Route::get('/solicitud/{ide}', [PaginasController::class, 'solicitud']);
//Route::get('/info-solicitud/{id}', 'PetController@getSolicitud');
Route::post('/crearSolicitud', [PetController::class, 'creaSolicitud']); ←

```

En la hoja de rutas nos redirige al PetController a la función creaSolicitud.

```

public function creaSolicitud(Request $r){
    $solicitud = new Solicitud();

    $solicitud->asuntoSolicitud = $r->tituloInput;
    $solicitud->idUsuarioC = $r->clienteInput;
    $solicitud->idUsuarioD = $r->liderInput;
    $solicitud->fechaSolicitud = $r->fechaSolicitudInput;
    $solicitud->idEstado = $r->estadoSolicitudInput;
    $solicitud->documentoDescripcion = ' ';

    request()->validate([
        'file' => 'required|mimes:doc,docx,pdf|max:5242880',
    ]);

    $solicitud->descripcionSolic = $r->descripcionInput;
    $solicitud->fechaInicio = $r->inicioInput;
    $solicitud->fechaFin = $r->terminoInput;

    $solicitud->retroalimentacion = " ";
    $solicitud->idDocumento = 1;

    $solicitud->save();
}

```

```
//Guardar el archivo después de haber registrado la solicitud para poder obtener el id y crear la carpeta del p
if ($r->hasFile('file')) {

    $path = $r->file('file')->storeAs(
        'Proyectos/'.$solicitud->id.'/Documentos/Solicitud', $r->file('file')->getClientOriginalName(),'public'
    );
    $url = Storage::url($path);

    $solicitud->documentoDescripcion=$url;
}

$solicitud->save();

return redirect('/solicitudes');
```

Se hace una nueva solicitud, después se guardan dentro de esta solicitud los datos que vienen en la request, dentro de la request se valida que el archivo que se haya subido corresponda a doc, docx y pdf con un tamaño máximo 5MB, al principio se guarda con un valor "" al documento descripción en caso de que la request tenga un archivo se guarda en la ruta que definimos para las solicitudes, después esa ruta la guardamos en la solicitud, la guardamos y redirigimos a la ruta solicitudes.

**Prueba 1**

Nombre del solicitante: Pancho Martinez Martinez  
 Solicitud enviada a: Eric Perez Perez

Fecha de solicitud: 2022-05-27  
 Fecha de inicio del proyecto: 2022-05-28  
 Fecha de termino del proyecto: 2022-06-10

Descripcion:  
 sisisisisis

**Estado**

Solicitud enviada

Revisar

Para poder revisar la solicitud y ver si hay algunos comentarios por parte del administrador acerca del proyecto tenemos que darle clic al botón verde que se ve ahí, una vez que le hemos dado clic nos va a aparecer lo siguiente.

Revisión de solicitud para el proyecto "Prueba 1"

Título/Nombre del proyecto  
 Prueba

Cliente  
 Pancho Martinez Martinez

Lider de proyecto  
 Eric Perez Perez

Descripción del proyecto  
 sisisisisis

Documento adjunto:

Descargar

Seleccionar archivo Sin archivos...eleccionados Reemplazar archivo de solicitud

Fecha de inicio  
 28/05/2022

Fecha de termino  
 10/06/2022

Ahí podemos ver los datos que rellenamos de la solicitud, y en caso de requerirlo podemos sustituir el documento que hayamos subido, o en su defecto ver el que subimos.

## Retroalimentación

Lider de proyecto Eric Perez Perez

Reenviar solicitud

Abajo nos aparecerá la retroalimentación del administrador, y en caso de que se rechace una solicitud aparecerá la razón de este rechazo, si no aparecerá como aprobada.

```
$(document).ready(function(){
    $(document).on('click','#btn',function(){
        var ide = $(this).val();
        $.ajax({
            type: "GET",
            url: "/solicitud/"+ide,
            success:function(data){
                $("#nav-solicitud").html(data);
            }
        });
    });
});
```

Obtenemos la solicitud una vez que demos clic en revisar, se envia el id por el botón y hacemos una petición de tipo get a la ruta de /solicitud/.

```
public function solicitud($ide){
    $solicitud=Solicitud::find($ide);

    return view('vistas-principal.revisionSolicitud',compact('solicitud'));
}
```

La función recibe un \$ide, hacemos una búsqueda en las solicitudes activas una que coincida con el id recibido, después se redirige a la vista de revisión de la solicitud con la solicitud que se haya encontrado.

```

<br>
<h5>Revisión de solicitud para el proyecto "{{solicitud->asuntoSolicitud}}"</h5>
<br>

<form @if(Auth::user()->id == $solicitud->UsuarioLider->id) action="/revisarSolicitud" @else action="/reenviarSolicitud"
@csrf
<div class="col">
    <div class="form-group">
        <input hidden type="number" class="form-control" name="idInput" id="idInput" value="{{solicitud->id}}>

        <label for="tituloInput2">Título/Nombre del proyecto</label>
        <input type="text" class="form-control" name="tituloInput" id="tituloInput" value="{{solicitud->asuntoSolicitud}}>
    </div>
</div>
<br>
<div class="row">
    <div class="col">
        <div class="form-group">
            <label for="clienteInput">Cliente</label>
            <select class="form-select" id="clienteInput" name="clienteInput" blocked>
                <option value="{{solicitud->UsuarioCliente->id}}>{{solicitud->UsuarioCliente->name}} {{solicitud->UsuarioCliente->id}}>
            </select>
        </div>
    </div>
</div>
<br>
<div class="row">
    <div class="col">
        <div class="form-group">
            <label for="liderInput">Lider de proyecto</label>
            <select class="form-select" id="liderInput" name="liderInput" aria-label="Selección líder proyecto">
                <option value="{{solicitud->UsuarioCliente->id}}>{{solicitud->UsuarioLider->name}} {{solicitud->UsuarioLider->id}}>
            </select>
        </div>
    </div>
</div>

```

En la vista se muestra la información que obtuvimos con la consulta, como vemos esta en un form por lo que se puede ver y editar lo que pusimos.

Dentro del contenedor del proyecto podemos darle clic en ver proyecto, para ver la información, acceder a los documentos o las tareas de ingeniería.

#### Involucrados:

**Cliente:** Pancho Martinez Martinez

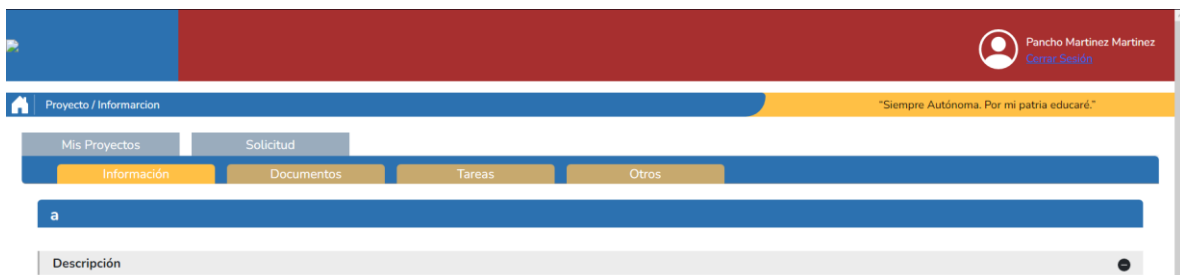
**Lider:** Eric Perez Perez

#### Descripción:

sesesese

[Ver proyecto](#)

Una vez entrando a esta sección del proyecto, podemos ver lo siguiente.





Podemos ver una sub barra de navegación donde está la información del proyecto, los documentos, las tareas de ingeniería y una llamada otros, esta es de prueba no se incluirá en la versión final.

En la pestaña de información podemos ver un desglose de la información referente al proyecto.

a

Descripción

sesesese

Fecha de inicio:2022-05-27

Fecha de termino:2022-06-02

Interesados y contactos

Cliente

Pancho Martinez Martinez

pancho@gmail.com

Involucrados

Lider

Eric Perez Perez

eric@gmail.com

Esta manera de acomodar el proyecto es un diseño con relación a la universidad por parte del equipo.

El apartado de documentos nos permite subir una etiqueta y un documento relacionado con está, la etiqueta nos permite identificar distintos tipos de documentos.

Documentos

Acta constitutiva

Agregar documento

UASLP

Universidad Autónoma de San Luis Potosí

Alvaro Obregón #64, Col. Centro, C.P. 78000

San Luis Potosí, S.L.P., México

Tel. +52 (444) 8262300

Alvaro Obregón #64, Col. Centro, C.P. 78000

San Luis Potosí, S.L.P., México

Tel. +52 (444) 8262300

Agregar nuevo documento

Etiqueta/Nombre del documento

Ingresa el nombre del documento

Agrega documento

Seleccionar archivo

Sin archivos...eleccionados

Enviar

Para enviar el formulario con la etiqueta y el documento, utilizamos el siguiente script.

```
//Funcion para agregar una nueva solicitud sin tener qu
$("#modal-formulario").submit(function(e) {
    e.preventDefault(); // avoid to execute the actual

    var formData = new FormData(this);
    $.ajax({
        type: 'POST',
        url: "{ url('/crearDocumento') }",
        data: formData,
        cache: false,
        contentType: false,
        processData: false,
        success: (data) => {
            $('#solicitudProyectoModal').modal('hide');
            $("#nav-documento").html(data);
        }
    });
});
```

Se hace una petición post, a la ruta de /creardocumento en nuestra hoja de rutas, se envía el formData.

```
function creaDocumento(Request $r){
    $d = new Documentos();
    $d->etiqueta = $r->tituloInput;
    $d->nombre = $r->file('file')->getClientOriginalName();

    $d->folioProyecto = $r->id;

    if ($r->hasFile('file')) {
        request()->validate([
            'file' => 'required|mimes:doc,docx,pdf|max:5242880',
        ]);

        $path = $r->file('file')->storeAs(
            'Proyectos/' . $r->id . '/' . 'Documentos/' . $r->tituloInput, $r->file('file')->getClientOriginalName(), 'publ
        );
        $url = Storage::url($path);

        $d->ruta=$url;
    }

    $d->save();
    $proyecto = Proyecto::find($r->id);
    return view('vistas-proyecto/documentosP', compact('proyecto'));
}
```

En esta función se crea un nuevo documento, se guarda la etiqueta y el nombre original del documento, así como el folio del proyecto para relacionar los documentos con el proyecto al que son subidos, si la petición tiene un archivo se valida que el archivo sea del tipo doc, docx o pdf y que sea menor a 5MB. Se almacena el archivo en la ruta Proyectos/id del proyecto/Documentos/ Etiqueta/ Nombre del Archivo/ public después

esta ruta se guarda en la variable ruta de los documentos, se guarda el documento y se regresa la vista de documentosP con el id del proyecto al que pertenece el documento.

```
function cargaDocumento(id){
    $.ajax({
        url: "/documento/" + id,
        success: function(data){
            $("#nav-plantilla").html(data);
        }
    });
}
```

Para poder ver el documento y la etiqueta que acabas de subir utilizamos esta función de carga documento.

```
public function documento($id){
    $documento = Documentos::find($id);
    return view('vistas-proyecto/vistas-documentos/plantillaD', compact('documento'));
}
```

En esta función se recibe el id del documento, se realiza una búsqueda de todos los documentos y se regresa con la vista de documentos a la plantillaD.

En la sección de documentos se pueden hacer comentarios referentes a los documentos por cualquier integrante del equipo y se verían de la siguiente manera.

Comentarios

Ing. Pancho Martinez Martinez  
hola

Comentar

Los comentarios se encuentran en un formulario de tipo Post, se le envían el id del documento y el del usuario para asociarlo con estos dos.

```
<form action="/publicarComentario" id="comentario" method="POST" enctype="multipart/form-data">
    @csrf
    <input hidden type="number" class="form-control" id="idDocumento" name="idDocumento" value="{{ $documento->id }}">
    <input hidden type="number" class="form-control" id="idUserario" name="idUserario" value="{{ Auth::user()->id }}">
    <textarea class="form-control" id="comentario" name="comentario" rows="3"></textarea>
    <br>
    <button type="submit" class="btn btn-primary">Comentar</button>
</form>
```

En el controlador tenemos la función publicarComentario que recibe la petición, crea un modelo del comentario y se agrega la información de la petición al modelo, se guarda y se busca el documento basándonos en la id del documento, después se le redirige a la ruta proyecto con el folio del proyecto al que se le hizo el comentario.

```
function publicarComentario(Request $r){
    $c = new Comentario();
    $c->idUsuario = $r->idUsuario;
    $c->idDocumento = $r->idDocumento;
    $c->contenido = $r->comentario;

    $c->save();

    $documento = Documentos::find($r->idDocumento);
    return redirect()->route('proyecto', [$documento->folioProyecto]);
}
```

Por último, en la pestaña de tareas, para poder crear una se tiene que dar clic en el botón de nueva tarea, se abrirá un modal donde rellenaremos los datos de la tarea como lo son el título, el encargado, la descripción de la tarea y una fecha de termino aproximada de la tarea. Así como el estatus de la tarea cuyas opciones son por hacer, en proceso, terminada. La lista de tareas se puede mover por las diferentes filas de la lista, dependiendo de la cantidad de tareas en cada lista se cambian los porcentajes que se encuentran en la vista del proyecto, esto aun no se encuentra implementado, pero es algo que se va a implementar posteriormente.

Proyecto / Tareas "Siempre Autónoma. Por mi patria educaré."

Mis Proyectos Solicitud Tareas Otros

Lista de Tareas

Nueva Tarea (+)

Analizar	Encargado	Descripción	Caducidad	Status
hacer pruebas	Eric Perez Perez	se analizan las pruebas	2022-06-03	En proceso
Diseñar	Encargado	Descripción	Caducidad	Status
Desarrollar	Encargado	Descripción	Caducidad	Status
Evaluar	Encargado	Descripción	Caducidad	Status

**UASLP**

Universidad Autónoma de San Luis Potosí  
Álvaro Obregón #64, Col. Centro, C.P. 78000  
San Luis Potosí, S.L.P., México  
Tel. +52 (444) 8262300

**Facultad de ingeniería**

Universidad Autónoma de San Luis Potosí  
Álvaro Obregón #64, Col. Centro, C.P. 78000  
San Luis Potosí, S.L.P., México  
Tel. +52 (444) 8262300

**Departamento de informática**

Universidad Autónoma de San Luis Potosí  
Álvaro Obregón #64, Col. Centro, C.P. 78000  
San Luis Potosí, S.L.P., México  
Tel. +52 (444) 8262300

Para cambiar el estado de una tarea, utilizamos la siguiente función:

```
function cambiaEstado(event, ide){
    var selectElement = event.target;
    var value = selectElement.value;

    if(selectElement.value == 8)
        selectElement.style.background = "#b53636";
    else if(selectElement.value == 9)
        selectElement.style.background = "#b58736";
    else if(selectElement.value == 10)
        selectElement.style.background = "#41b536";


    $.ajax({
        type: "POST",
        url: '/cambiarEstadoTarea',
        data: {"_token": "{{ csrf_token() }}", id: ide, estado: value},
        success: function()
        {
        }
    });
}
```

En la que recibe el evento de cambia estado y el ide de la tarea, tenemos dos variables, una en la que tenemos el elemento que seleccionamos y el valor que tiene este elemento en su estado, dependiendo del valor del estado se cambia el color del fondo del estado. Por hacer es 8, En proceso es 9 y Terminada es 10.

```
<table class="table-sortable">
    <thead>
        <tr>
            <th class="titulo-tabla" id="titulo-analizar" scope="col">Analizar</th>
            <th scope="col">Encargado</th>
            <th scope="col">Descripción</th>
            <th scope="col">Caducidad</th>
            <th scope="col">Status</th>
        </tr>
    </thead>
    <tbody class="container-list" id="lista-analizar">
        @foreach($proyecto->Tareas as $t)
            @if($t->Fase->nombreEstado == 'Analizar')
                <tr id="{{ $t->id }}">
                    <th class="col-md-2 titulo-tarea" scope="row">{{ $t->titulo }}</th>
                    <td class="col-md-2">{{ $t->Usuario->name }} {{ $t->Usuario->apellidoPaterno }} {{ $t->Usuario->apellidoMaterno }}</td>
                    <td class="col-md-4">{{ $t->descripcion }}</td>
                    <td class="col-md-1">{{ $t->fechaFinal }}</td>
                    <td class="col-md-2">
                        <select onchange="cambiaEstado(event,{{ $t->id }})" class="form-select" id="encargadoInput" name="encargado">
                            <option value=8 @if($t->Estado->nombreEstado == 'Por hacer') selected @endif>Por hacer</option>
                            <option value=9 @if($t->Estado->nombreEstado == 'En proceso') selected @endif>En proceso</option>
                            <option value=10 @if($t->Estado->nombreEstado == 'Terminada') selected @endif>Terminada</option>
                        </select>
                    </td>
                </tr>
            @endif
        @endforeach
    </tbody>
</table>
```

Las listas de tareas están separadas por secciones, las cuales muestran las tareas mediante un foreach donde se muestra una fila con los detalles de la tarea, así como el estado de esta con su color correspondiente.

Dentro del proyecto hay una sección para el administrador, donde este podrá ver las solicitudes de registro que se tengan, los usuarios activos y la opción de agregar un usuario. En la sección se planea implementarla posteriormente.



admin

Cerrar Sesión

Inicio / Panel Administrativo

"Siempre Autónoma. Por mi patria educaré."

Mis Proyectos

Solicitud

Admin Panel



Usuarios

Administrar solicitudes

Administrar usuarios

Agregar usuario

Lista de usuarios registrados

Usuarios registrados				
#	Nombre	Apellidos	Correo	Eliminar cuenta
2	Pancho	Martinez Martinez	pancho@gmail.com	
3	Eric	Perez Perez	eric@gmail.com	

UASLP

Universidad Autónoma de San Luis Potosí  
Álvaro Obregón #64, Col. Centro, C.P. 78000  
San Luis Potosí, S.L.P., México  
Tel. +52 (444) 8262300

Facultad de ingeniería

Universidad Autónoma de San Luis Potosí  
Álvaro Obregón #64, Col. Centro, C.P. 78000  
San Luis Potosí, S.L.P., México  
Tel. +52 (444) 8262300

Departamento de informática

Universidad Autónoma de San Luis Potosí  
Álvaro Obregón #64, Col. Centro, C.P. 78000  
San Luis Potosí, S.L.P., México  
Tel. +52 (444) 8262300