ASISTENCIAPP (VERSION BETA)

INGENERIA DE SOFTWARE II.

Desarrolladores: ***Fredy Mendoza Vargas y Luis Giovanny Carreño.***

Supervisor: ***Jairo Enrique Serrano Casteñada.***

Un Sistema para Regular el control de asistencia y permanencia de los estudiantes en las clases y Facilitarle a los docentes una Herramienta Tecnológica (A través de una Aplicación) a la labor efectiva de trabajo pedagógico de los docentes de cualquiera de las Instituciones Educativas.

TABLA DE CONTENIDO

Contenido

Introducción:

En este documento se explicara claramente la idea de una aplicación que permita controlar y administrar la asistencia a clases de los estudiantes en cualquier institución educativa. También para tener una idea general del entorno por el cual nació la idea de realizar esta aplicación.

Justificación

Este Proyecto está fundamentado en la necesidad de Regular el control de asistencia y permanencia de los estudiantes en las clases y Facilitarle a los docentes una Herramienta Tecnológica (A través de una Aplicación) a la labor efectiva de trabajo pedagógico de los docentes de cualquiera de las Instituciones Educativas.

Antecedente o definición del problema:

Los docentes para tener una labor efectiva pedagógica, toman asistencias a los estudiantes de sus cursos asignados para autoevaluarse e incluso para controlar también la permanencia de sus estudiantes. Pero se tenía un control muy precario (poca estabilidad), puesto que realizaban sobre papel y generalmente la información suministrada era muy escasa y se dificultaba ver un historial en la planilla de asistencia. La herramienta "Excel", Permite que los docentes sistematicen y lleven un control, pero aun así este proceso requiere de tiempo de las horas de clase establecida, y en otro caso, el observar de forma estadística de quien asista a clases o no, tampoco no es muy amigable y requiere también de mucho tiempo para el docente. Por estos motivos nace la necesidad de poder realizar estos procesos de control de asistencia de manera rápida y segura, donde la información siempre se encuentra disponible y permita llevar el control adecuado de los estudiantes a cargos del profesor.

Objetivo

Desarrollar una aplicación de software que permita controlar la asistencia y permanencia de los estudiantes en las clases y Facilitarle a los docentes una Herramienta Tecnológica amigable, confiable y segura, donde puedan tomar la asistencia rápida y ver los registros de asistencia de una forma visualmente de los estudiantes individualmente y de todo un grupo de estudiantes de una clase.

Análisis de requerimientos

Este sistema fue validado para la clase de ingeniería del Software II en donde se establecieron los siguientes requerimientos:

GENERALIDADES

Este sistema se implementara para la gestionar las asistencias a cada uno de los cursos impartidos dentro de la universidad Tecnológica de Bolívar.

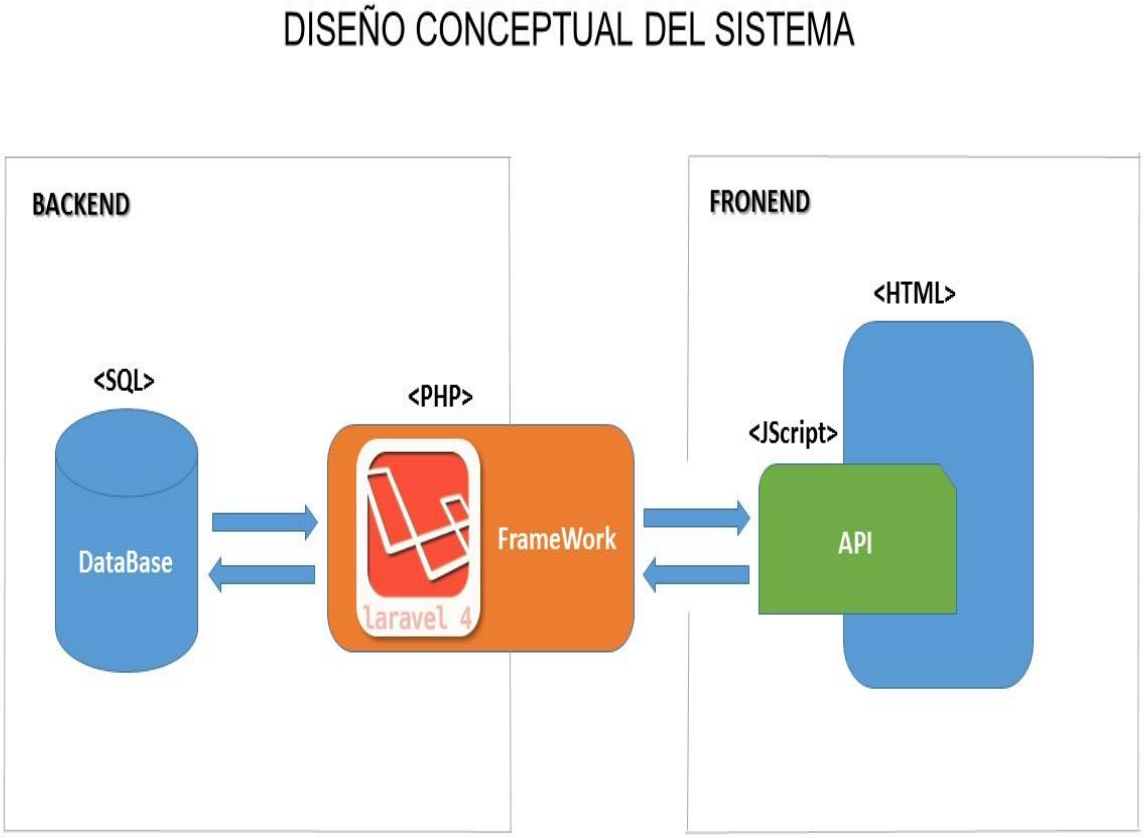
REQUERIMIENTOS

- *El sistema debe de identificar a través de un proceso de autenticación a cada uno de los profesores que tengan cursos asignados.*
- *Dentro del sistema se debe de asignar los estudiantes pertinentes de cada grupo a un profesor determinado.*
- *Cada profesor puede tomar asistencias de sus clases y estas serán registradas en el sistema.*
- *Los profesores solo pueden tomar asistencia dentro de las horas estipuladas del curso y dentro de un intervalo considerable.*
- *El sistema debe tener la capacidad de mostrar reportes por día de cada una de las asistencias tomas por los profesor*
- *El sistema debe de generar alarmas para insistencias concurrentes por un estudiante*

HISTORIA DE USUARIOS.

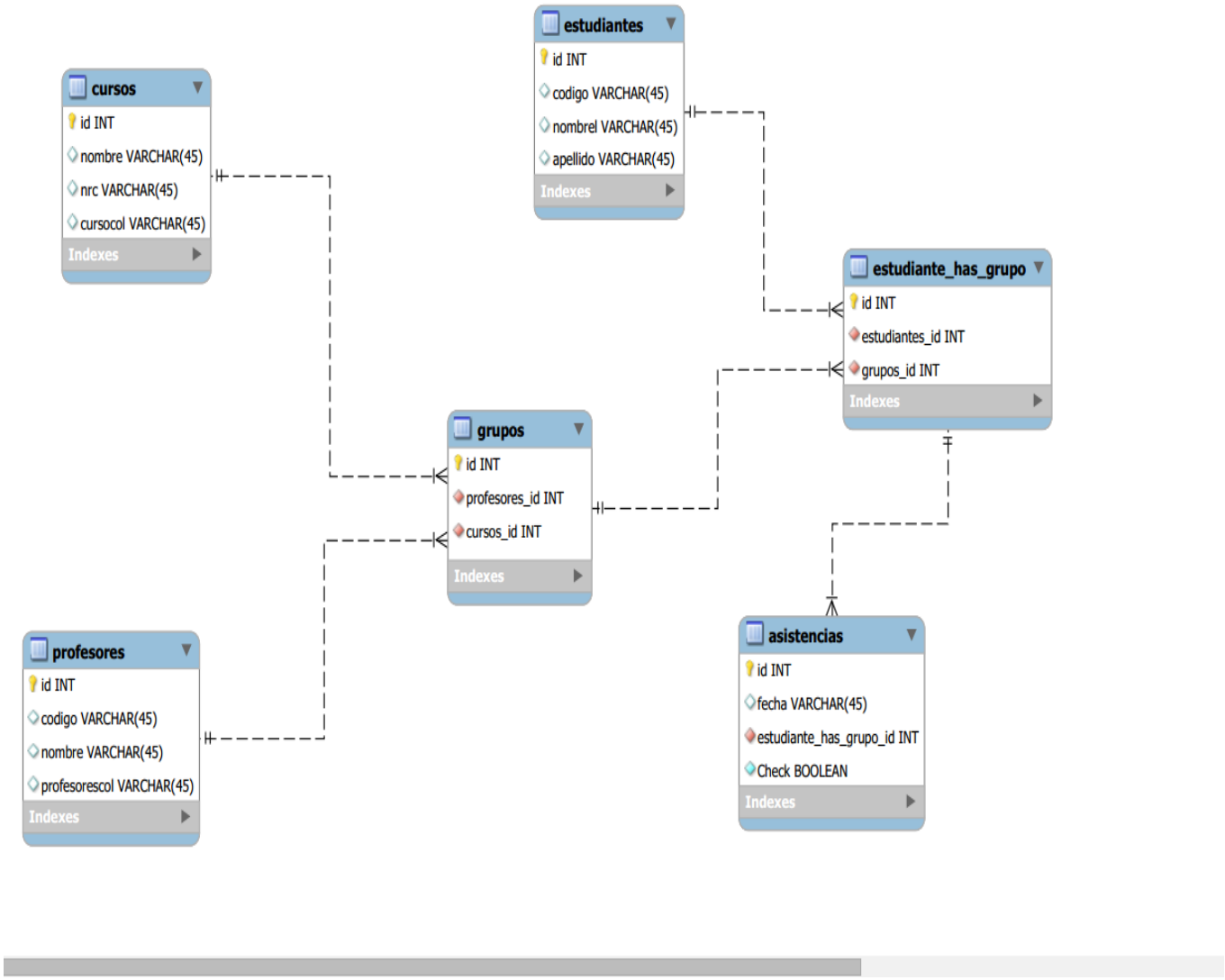
A continuación se presentan la historia de usuarios realizados en la clase de software II en la Universidad Tecnológica de Bolívar en el salón A1-401. Y se llegaron a las conclusiones:

Diseño Conceptual del Sistema.



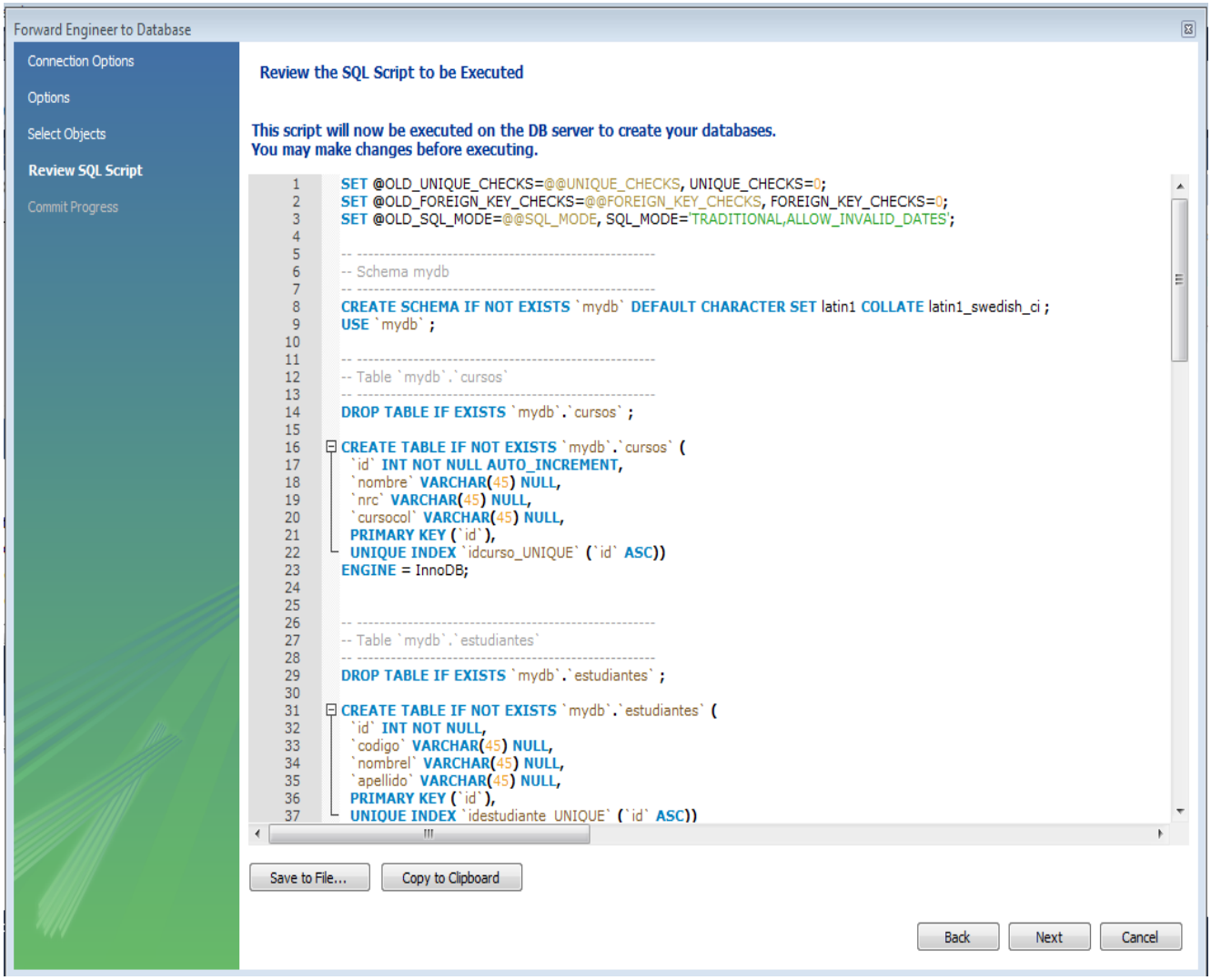
Modelo de datos.

El Modelo de datos que se va a necesitar para El Almacenamiento y Consulta de Información de la aplicación.



Nota: el Modelo de dato fue diseñado el MySQL Workbrench

Ahora cuando le exportamos el modelo realizado en MySQL Workbench nos muestra los comandos en SQL para el uso, creación y manejo de las tablas de la base de datos.



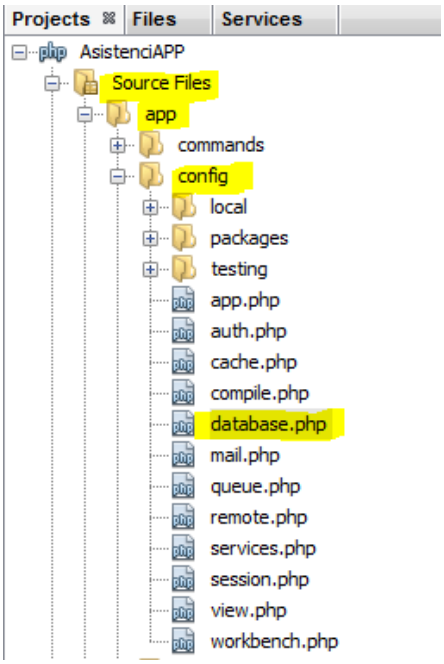
Implementación en el Backend.

Luego de Tener nuestra base de datos. Procedemos a la implementación del backend. El backend fue implementado con el PHP Framework Laravel 4. Le mostraremos como fue desarrollado de una manera rápida y directa la creación de los modelos, controladores, rutas y la configuración con la conexión de la base de datos. Para más información de cómo crear un proyecto en laravel visite: *http://laravel.com/docs/installation*

CONFIGURACIÓN DE LA CONEXIÓN DE LA BASE DE DATOS.

Al crear nuestro proyecto configuramos la base de datos en el archivo de configuración de database.php del proyecto de laravel; esto es para que nuestros modelos y controladores se comuniquen con nuestra base de dato ya creada. Entonces procedemos:

- Tenemos el proyecto abierto en netbeans. Nos vamos a la carpeta *Source File/ app/ config/database.php.*



- Comprobamos que esté por defecto el motor de base de datos en la que tenemos nuestra base de datos. *'default' => 'mysql',*

```
/*
|--------------------------------------------------------------------------
| Default Database Connection Name
|--------------------------------------------------------------------------
|
| Here you may specify which of the database connections below you wish
| to use as your default connection for all database work. Of course
| you may use many connections at once using the Database library.
|
*/

'default' => 'mysql',
```

- Procedemos a realizar la conexión en: host => localhost, database => <<nombre de la base de datos>>, username => <<nombre de usuario de la conexión>>, password => <<Contraseña de la conexión>> y salvamos.

```
'connections' => array(

        'sqlite' => array(
                'driver'   => 'sqlite',
                'database' => __DIR__.'/../database/production.sqlite',
                'prefix'   => '',
        ),

        'mysql' => array(
                'driver'    => 'mysql',
                'host'      => 'localhost',
                'database'  => 'mydb',
                'username'  => 'root',
                'password'  => 'pass',
                'charset'   => 'utf8',
                'collation' => 'utf8_unicode_ci',
                'prefix'    => '',
        ),
```

IMPLEMENTACION DE LOS MODELOS.

Modelos:

Los modelos son clases que representan las tablas de una base de datos. Además que esa clases tienen métodos que tendrán una serie de consulta (relaciones entre ellas) y cada objeto instanciado sería un registro *[1]*.

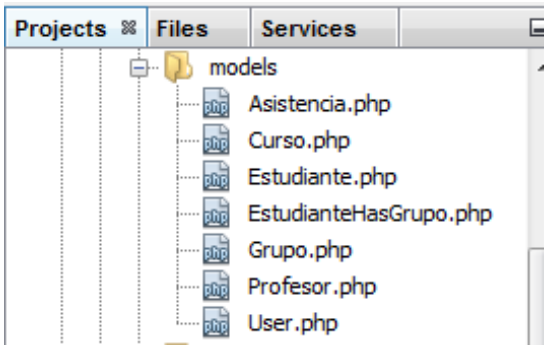
Eloquents:

Es una clase ORM (*Objec-Relational-Mapping*) de Laravel que nos permitirá solucionar el problema de tener que crear las consultas para la base de datos, además que estas consultas suelen repetirse entre tabla y tabla. Insertar registros, modificarlos, eliminarlos, buscar un registro por su id, listar registros, etc. Ya que esto es una de las cosas más densas pero necesarias para el desarrollo *[1]*.

> ***ORM:* Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional, utilizando un motor de persistencia *[2]***

Tomado en Línea en: [1] http://fernando-gaitan.com.ar/laravel-parte-5-modelos/ [2] http://es.wikipedia.org/wiki/Mapeo_objeto-relacional

La implementación de los modelos se encuentra dentro de: Source *File/ app/models/…* A continuación se presentaran cada uno de ellos:



Asistencia.php

```php
<?php

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

class Asistencia extends Eloquent{
    protected $table = 'asistencias';
    public $timestamps = false;

    public function estudianteHasGrupo(){
        return $this->belongsTo('EstudianteHasGrupo');
    }

}
```

Como pudimos observar en el modelo de datos. La entidad asistencia, que en este caso en los modelos son objetos, el modelo Asistencia tiene relacion con EstudianteHasGrupo y tiene como atributo la tabla asistencias de la DB.

Curso.php

```php
<?php

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

class Curso extends Eloquent{
    protected $table = 'cursos';
    public $timestamps = false;

    public function grupo(){
        return $this->hasMany('Grupo');
    }

}
```

Como pudimos observar en el modelo de datos. el modelo Curso tiene relacion con Grupo y tiene como atributo la tabla cursos de la DB.

Estudiante.php

```php
class Estudiante extends Eloquent{

    protected $table = 'estudiantes';
    public $timestamps = false;

    public function estudianteHasGrupo(){
        return $this->hasMany('EstudianteHasGrupo');
    }
}
```

Como pudimos observar en el modelo de datos. El modelo Estudiantes tiene relacion con Grupo y tiene como atributo la tabla estudiantes de la DB.

EstudianteHasGrupo.php

```php
class EstudianteHasGrupo extends Eloquent{

    protected $table = 'estudiante_has_grupo';
    public $timestamps = false;

    public function grupo(){
        return $this->belongsTo('Grupo');
    }

    public function estudiante(){
        return $this->belongsTo('Estudiante');
    }

    public function asistencia(){
        return $this->hasMany('Asistencia');
    }
}
```

Como pudimos observar en el modelo de datos. El modelo EstudiantesHasGroup tiene relacion con Grupo,Estudiante, Asistencia y tiene como atributo la tabla estudiantes_has_grupo de la DB.

Grupo.php

```php
class Grupo extends Eloquent{
    protected $table = 'grupos';
    public $timestamps=false;

    public function curso(){
        return $this->belongsTo('Curso');
    }

    public function profesor(){
        return $this->belongsTo('Profesor');
    }

    public function estudianteHasGrupo(){
        return $this->hasMany('EstudianteHasGrupo');
    }
}
```

Como pudimos observar en el modelo de datos. El modelo Group tiene relacion con Curso, Profesor, EstudianteHasGrupo y tiene como atributo la tabla grupos de la DB.

Profesor.php

```php
class Profesor extends Eloquent{
    protected $table = 'profesores';
    public $timestamps = false;

    public function grupo(){
        return $this->hasMany('Grupo');
    }

}
```

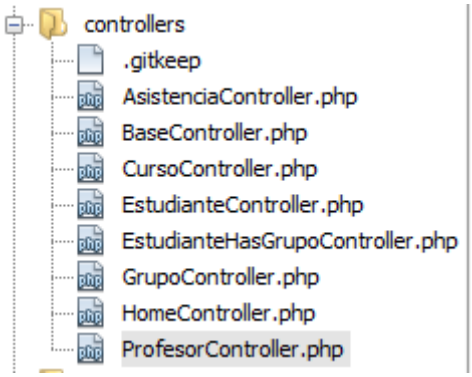
Como pudimos observar en el modelo de datos. El modelo Profesor tiene relacion con Grupo y tiene como atributo la tabla profesores de la DB

IMPLEMENTACION DE LOS CONTROLADORES.

Controladores:

Los controladores representan una parte de nuestra aplicación representados con clases, que a su vez estarán divididos en acciones, representados como métodos.

La implementación de los Controladores se encuentra dentro de: Source *File/ app/controllers/...* A continuación se presentaran cada uno de ellos:



AsistenciaController.php

```
class AsistenciaController extends \BaseController{
    /**
    * Display a listing of the resource.
    * @return Response
    */
   public function index() {
      $asistencias = Asistencia::all()->toJson();
      return $asistencias;
   }
```

```
    public function show($idAsistencia) {
        $asistencia = Asistencia::find($idAsistencia)->toJson();
        return $asistencia;
    }

    public function create(){
        $hasgrupo = EstudianteHasGrupo::all();
        $grupoId=Input::get('grupo_id');
        if($hasgrupo->grupos_id ===$grupoId){
            $asistencia = new Asistencia;
            $asistencia->fecha = date("Y/m/d/H/i/s");
            $asistencia->estudiante_has_grupo_id = $hasgrupo->id;
            $asistencia->Check = true;
            $grupo=  EstudianteHasGrupo::find($hasgrupo->id);
            $asistencia->$grupo->associate($grupo);
            $asistencia->save();
            return "Guardado de las Asistencia Fue Con Exito.";
        }
    }

}
```

Como podemos observar el controlador tiene tres funciones que son: ***index()*** que es aquella cuando le damos la ruta (que se verá más adelante) nos va a retornar un JSON de todos los datos en la base de datos por medio del modelo Asistencia. La función ***show($id)*** retorna un JSON de la información de uno Particular por medio del $id. Y la Función ***créate()*** que es la función que almacena en la base de datos la asistencia al recibir un Ajax.

*Curso*Controller*.php*

```php
class CursoController extends \BaseController{

    /**
     * Display a listing of the resource.
     * @return Response
     */

    public function index() {
        $cursos = Curso::all()->toJson();
        return $cursos;
    }

    /**
     * Display the specified resource.
     * @param  int  $id
     * @return Response
     */
    public function show($idCurso) {
        $curso = Curso::find ($idCurso)->toJson();
        return $curso;
    }
}
```

Como podemos observar el controlador tiene dos funciones que son: ***index()*** que es aquella cuando le damos la ruta nos va a retornar un JSON de todos los datos en la base de datos por medio del modelo Curso. La función ***show($id)*** retorna un JSON de la información de uno Particular por medio del $id.

EstudianteController.php

```php
class EstudianteController extends \BaseController{

    /**
     * Display a listing of the resource.
     * @return Response
     */
    public function index() {
        $estudiantes = Estudiante::all()->toJson();
        return $estudiantes;
    }

    /**
     * Display the specified resource.
     * @param  int  $id
     * @return Response
     */
    public function show($idEstudainte) {
        $estudiante = Estudiante::find ($idEstudainte)->toJson();
        return $estudiante;
    }
}
```

Como podemos observar el controlador tiene dos funciones que son: ***index()*** que es aquella cuando le damos la ruta nos va a retornar un JSON de todos los datos en la base de datos por medio del modelo Estudiante. La función ***show($id)*** retorna un JSON de la información de uno Particular por medio del $id.

EstudianteHasGrupoController.php

```
class EstudianteHasGrupoController extends \BaseController{

    /**
     * Display a listing of the resource.
     * @return Response
     */
    public function index() {
        $estudianteHasGrupo = EstudianteHasGrupo::all()->toJson();
        return $estudianteHasGrupo;
    }

    /**
     * Display the specified resource.
     * @param  int  $id
     * @return Response
     */
    public function show($idEstudianteHasGrupo) {
        $estudianteHasGrupo = EstudianteHasGrupo::find($idEstudianteHasGrupo)->toJson();
        return $estudianteHasGrupo;
    }
}
```

Como podemos observar el controlador tiene dos funciones que son: ***index()*** que es aquella cuando le damos la ruta nos va a retornar un JSON de todos los datos en la base de datos por medio del modelo EstudianteHasGrupo. La función ***show($id)*** retorna un JSON de la información de uno Particular por medio del $id.

GrupoController.php

```php
class GrupoController extends \BaseController{

    /**
     * Display a listing of the resource.
     * @return Response
     */
    public function index() {
        $grupos = Grupo::all()->toJson();
        return $grupos;
    }

    /**
     * Display the specified resource.
     * @param  int  $id
     * @return Response
     */
    public function show($idGrupo) {
        $grupo = Grupo::find ($idGrupo)->toJson();
        return $grupo;
    }
}
```

Como podemos observar el controlador tiene dos funciones que son: ***index()*** que es aquella cuando le damos la ruta nos va a retornar un JSON de todos los datos en la base de datos por medio del modelo Grupo. La función ***show($id)*** retorna un JSON de la información de uno Particular por medio del $id.

ProfesorController.php

```php
class ProfesorController extends \BaseController {

    /**
     * Display a listing of the resource.
     * @return Response
     */
    public function index() {
        $profesores = Profesor::all()->toJson();
        return $profesores;
    }

    /**
     * Display the specified resource.
     * @param  int  $id
     * @return Response
     */
    public function show($id) {
        $profesor = Profesor::find ($id)->toJson();
        return $profesor;
    }

}
```

Como podemos observar el controlador tiene dos funciones que son: ***index()*** que es aquella cuando le damos la ruta nos va a retornar un JSON de todos los datos en la base de datos por medio del modelo Profesor. La función ***show($id)*** retorna un JSON de la información de uno Particular por medio del $id.

RUTAS.

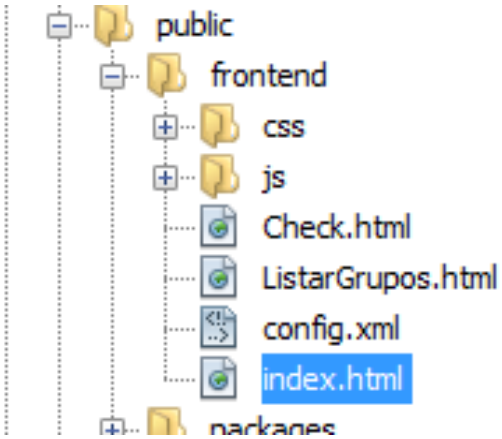
Ya con el controlador listo con todas las acciones que necesitamos vamos a pasar a crear las rutas para acceder a esas acciones, para esto nuestro archivo routes.php debe quedar de la siguiente manera:

```
Route::get('/', function()
{
        return View::make('hello');
});

Route::resource('profesor', 'ProfesorController');

Route::resource('grupo', 'GrupoController');

Route::resource('estudiante', 'EstudianteController');

Route::resource('curso','CursoController');

Route::resource('estudiante','EstudianteController');

Route::resource('estudiantehasgrupo', 'EstudianteHasGrupoController');

Route::resource('asistencia','AsistenciaController');
```

Ya con haber definido las rutas podemos concluir con nuestro backend de la Aplicación.

Implementación en el Fronted.

La implementación del frontend, para la prueba se encuentra en el mismo proyecto de laravel dentro de la carpeta public/frontend/. Hay encontramos todos los archivos HTML, CSS y Javascript, que para el alcance del proyecto es colocarlo en la carpeta www de un proyecto con apache cordova.



El frontend que fue implementado usando:

- HTML, CSS y JavaScript
- Jquery Mobile.
- Apache Cordova (IDE Netbeans 8.0)

Conclusión del Aprendizaje

EL proyecto de desarrollo ASISTENCIAPP nos permitió obtener competencias acerca del desarrollo software de una forma clara y precisa. En este proceso se tomó la investigación como eje principal, por lo cual, se conocieron todas las diferentes metodologías de desarrollo de software actuales y las herramientas tecnológicas fundamentales más utilizadas y de gran impacto dentro del entorno.

El proyecto se desarrolló bajo el proceso de metodología ágil en la cual fue fundamental el análisis de los requerimientos básicos, los cuales fueron bases fundamentales para poder estipular funcionabilidades básicas las cuales dan pie para el inicio del desarrollo general.

El enriquecimiento tecnológico dentro de este desarrollo es sobresaliente, cabe desatacar, que se realizaron introducciones a muchas de las herramientas tecnológicas las cuales son de gran envergadura dentro de la industria de software. Para este procese se utilizaron varias tales como Phonegap, Laravel Framework y Git entre otras, por lo cual el manejo y compresión de nuevas formas de desarrollo ha sido incentivadas en nuestra formación, haciendo mucho más interesante la elaboración de este proyecto.

El modelamiento de datos es parte fundamental para el desarrollo, ya que este fundamenta las base del modelo y determina ciertas funcionalidades del sistema, esta elaboración tomo como base primordial el modelo de datos, lo cual fue de gran ayuda para el resto del sistema, tener en cuenta e identificar qué información se debe procesar y por quien se debe procesar fundamenta los principios del MVC (Modelo Vista Controlador) el cual fue implementado en este desarrollo.

Por lo tanto esta forma de elaborar proyectos de software busca simplificar y optimizar actividades cotidianas dentro del desarrollo de software brindando y mejorando las capacidades para obtener mejores resultados en cada uno de los proyectos a desarrollar.

Descargar La Aplicación.

Para probar la aplicación y ver el código puedes descargarlo en:

https://github.com/fremeva/AsistenciAPP

Encontramos Todas las diferentes versiones y Archivos para la realización del proyecto.

Para correr y ver el código *(sugerimos el entorno de desarrollo Netbeans)* El proyecto se encuentra en la carpeta proyectoNetbeans/AsistenciApp.

Muchas Gracias.