**\*\*\* Documentación Ingeteam \*\*\***

Contenido

[1. Problemas planteados y cómo solucionarlos. 1](#_Toc63651653)

[1.1 Análisis del problema y soluciones. 2](#_Toc63651654)

[2. Creación de los diagramas de la base de datos. 2](#_Toc63651655)

[3. Creación de la base de datos. 3](#_Toc63651656)

[4. Creación del proyecto 4](#_Toc63651657)

[5. Proyecto en Laravel: 5](#_Toc63651658)

[5.1 Creación de los modelos. 5](#_Toc63651659)

[5.2 Relación de los modelos. 6](#_Toc63651660)

[5.3 Creación de controladores, vistas, rutas… 6](#_Toc63651661)

[6. Proyecto en Symfony: 7](#_Toc63651662)

[6.1 Creación de los modelos. 7](#_Toc63651663)

[6.2 Relación de los modelos. 8](#_Toc63651664)

[6.3 Validación de campos para los formularios. 8](#_Toc63651665)

[6.4 Registro, Login y Logout 8](#_Toc63651666)

[6.5 CRUD de las tareas 9](#_Toc63651667)

[6.6 Accesos de control 10](#_Toc63651668)

[7. Dependencias 10](#_Toc63651669)

# Problemas planteados y cómo solucionarlos.

Antes de llevar a cabo la iniciación de un proyecto o bien la modificación de uno ya existente lo más importante es analizar detenidamente el problema, atender las peticiones y entender qué es lo que quiere el cliente y, a partir de ahí, buscar una solución.

En este caso, el departamento de Producto nos está pidiendo una modificación al proyecto ya existente pero como es una prueba y no tenemos proyecto, haremos uno desde 0.

## 1.1 Análisis del problema y soluciones.

Bien. Si leemos lo que el departamento nos está pidiendo es lo siguiente:

Quieren una pequeña aplicación que incluya el uso de usuarios (por lo tanto, registro, login y logout de los mismos) y también el uso de tareas (un CRUD)

También nos pide que estas tareas sean **únicamente** visibles, editables y borrables por el propio usuario que las crea.

A esto vamos a añadir una *pequeña modificación* y es que nos vamos a encargar también de que tengan un usuario administrador que pueda visibilizar las tareas de **todos** los usuarios así como editarlas y borrarlas. También podrá crear tareas para otros usuarios.

Además de esto, añadiremos una restricción. Y es que los usuarios únicamente pueden crear tareas para ellos mismos. Tampoco podrán modificar el usuario que ha creado la tarea y asignarle otra. Cosa que el administrador **sí podría.**

# Creación de los diagramas de la base de datos.

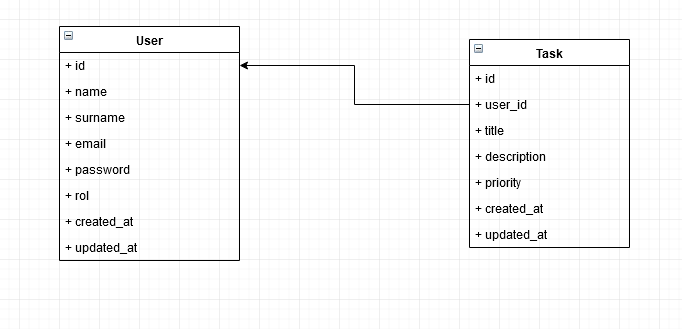
Bien. Una vez teniendo una solución en mente para el problema planteado es hora de realizar los diagramas.

En este caso, al ser una pequeña modificación o un pequeño proyecto será muy simple.

El Modelo constará de dos tablas.

Una de ellas será la tabla de usuarios que se encargará de registrar y loguear usuarios.

La otra será la de tareas. Que servirá para almacenar las tareas.



La tabla Task tendrá una clave foránea de la tabla de usuario para así saber a quién pertenece cada una de las tareas.

Las propiedad *created\_at* será un *datetime* de cuando el usuario o la tarea fueron dados de alta en la base de datos, mientras que *updated\_at* (que también será un *datetime*) sirve para reflejar la última actualización de un registro.

# Creación de la base de datos.

Ya teniendo el driagrama completado, solo nos queda crear la base de datos.

Archivo [SQL](database.sql)

CREATE DATABASE IF NOT EXISTS ingeteam;

use test\_ingeteam;

CREATE TABLE IF NOT EXISTS users (

id int(255) not null auto\_increment,

name varchar(50) not null,

surname varchar(50) not null,

email varchar(255) not null,

password varchar(255) not null,

rol varchar(5) not null,

image varchar(255),

created\_at datetime,

updated\_at datetime,

CONSTRAINT pk\_id PRIMARY KEY (id)

)ENGINE=InnoDb;

CREATE TABLE IF NOT EXISTS tasks (

id int(255) not null auto\_increment,

user\_id int(255) not null,

title varchar(100) not null,

description text not null,

priority varchar(50) not null,

created\_at datetime,

updated\_at datetime,

CONSTRAINT pk\_id PRIMARY KEY (id),

CONSTRAINT fk\_user\_task FOREIGN KEY (user\_id) REFERENCES users(id)

)ENGINE=InnoDb;

# Creación del proyecto

A continuación vamos a crear el proyecto y a documentar paso por paso. En este caso, he decidido hacer el proyecto tanto en Laravel como en Symfony.

# Proyecto en Laravel:

Vamos a empezar con el proyecto en Laravel. Para eso primero tenemos que tener un entorno de desarrollo para PHP como LAMP, WAMP o XAMPP.

Yo voy a usar xampp, así que una vez creada la carpeta en htdocs vamos con ellos.

Creamos el proyecto en laravel:

composer create-project laravel/laravel "01-Laravel" --prefer-dist

Conectamos a la base de datos:

Para ello editamos el archivo .env y ponemos la base de datos que deseamos usar.

## 5.1 Creación de los modelos.

Ahora vamos a crear los modelos. Para ellos ejecutamos el comando:

php artisan make:model Task

Creamos solo el modelo Task. ¿Por qué? Bien. Laravel nos ofrece una maravillosa funcionalidad que es la posibilidad de implementar un Auth.

Un Auth es, la automatización de creación del modelo, controlador y métodos para el registro, logueo y logout de usuarios.

Para usar Auth:

composer require laravel/ui (Instala UI para poder usar el auth)

php artisan ui:auth (Crea el controlador, vistas, rutas... etc (tambien el home))

php artisan ui bootstrap (Para instalar bootstrap)

npm install && npm run dev (Para compilar el home, bootstrap... etc)

## 5.2 Relación de los modelos.

Para ver esto más a fondo recomiendo revisar el código de cada uno de los ficheros y también sus comentarios.

[Task](01-Laravel/app/Models/Task.php)

[User](01-Laravel/app/Models/User.php)

## 5.3 Creación de controladores, vistas, rutas…

php artisan make:controller UserController (generar controlador)

php artisan make:controller TaskController

Dentro de cada controlador tendremos métodos que cada uno estará asignado a una vista o una ruta.

Ver los Archivos:

TaskController: en \01-Laravel\app\Http\Controllers

Carpeta Auth: en \01-Laravel\app\Http\Controllers\Auth

Web (routes): en \01-Laravel\routes

Carpeta task (views) en: \01-Laravel\resources\views

Carpeta auth (views) en: \01-Laravel\resources\views

Ahí podrán encontrar todo el código con comentarios.

# Proyecto en Symfony:

También vamos a desarrollar el proyecto en symfony. Para ello creamos el proyecto:

composer create-project symfony/website-skeleton 02-Symfony

También tenemos que añadir el comando para Apache, sino no se active el mod rewrite.

composer require symfony/apache-pack

Conectamos a la base de datos:

Para ello editamos el archivo .env y descomentamos la base de datos mysql, ponemos el usuario y la base de datos. Comentamos la PostgreSQL ya que no la usaremos.

## 6.1 Creación de los modelos.

Para crear los modelos en Symfony, tenemos que ejecutar una serie de comandos.

php bin/console doctrine:mapping:import App\Entity annotation --path=src/Entity

(importa los modelos de la base de datos directamente a la app)

php bin/console make:entity --regenerate //Getters y setters

Con esto ya tendremos importados los modelos en nuestra app.

## 6.2 Relación de los modelos.

Para relacionar los modelos es muy sencillo.

En Task añadimos:

    @ORM\ManyToOne(targetEntity="App\Entity\User", inversedBy="tasks")

Y en User añadimos:

@ORM\OneToMany(targetEntity="App\Entity\Task", mappedBy="user")

Con esto ya tendríamos la relación hecha.

## 6.3 Validación de campos para los formularios.

Ahora hemos de validar los campos.

Tanto nombre como apellidos únicamente nos interesa que tengan letras.

El Email ha de ser un tipo email.

Y el resto de los campos (Contraseña y todos los de Task además de nombre, apellidos y email) no pueden estar en blancos.

En los archivos dejo toda la configuración y comentarios de código.

Modelo [Task](02-Symfony/src/Entity/Task.php)

Modelo [User](02-Symfony/src/Entity/User.php)

## 6.4 Registro, Login y Logout

A diferencia de Laravel, aquí no tenemos una automatización, hay que hacerlo todo a mano.

Tanto para el registro como para el login y logout hay que tocar el archivo security.

En el registro para añadir los encoders para el cifrado de Contraseña, y en el login y logout definir el path y en el caso del login, el login check y su provider.

También para el login estaremos añadiendo nuestro métodos en el Modelo del User.

Modelo [User](02-Symfony/src/Entity/User.php)

Fichero [Security](02-Symfony/config/packages/security.yaml)

Fichero [RegisterType](02-Symfony/src/Form/RegisterType.php) que incluye el formulario de registro

Fichero [UserController](02-Symfony/src/Controller/UserController.php) que incluye todos los métodos

[Vistas](02-Symfony/templates/user) del UserController

Fichero de [Rutas](02-Symfony/config/routes.yaml)

## 6.5 CRUD de las tareas

Como en el proyecto anterior también tenemos un CRUD de tareas que consiste en lo mismo. Los usuarios únicamente pueden ver, editar y borrar sus propias tareas. Solo pueden crear tareas para ellos mismos.

El administrador puede ver, editar y borrar las tareas de todos los usuarios. También puede crear tareas para todos ellos.

Tanto para crear como para editar tareas requerimos de un formulario al igual que con el registro.

Fichero [TaskCreateType](02-Symfony/src/Form/TaskCreateType.php) que incluye el formulario para crear tareas.

Fichero [TaskEditType](02-Symfony/src/Form/TaskEditType.php) que incluye el formulario para editar tareas.

Fichero [TaskController](02-Symfony/src/Controller/TaskController.php) que incluye todos los métodos de las tareas.

[Vistas](02-Symfony/templates/task) del TaskController

Fichero de [Rutas](02-Symfony/config/routes.yaml)

## 6.6 Accesos de control

También en el fichero [Security](02-Symfony/config/packages/security.yaml) tenemos que modificar los accesos. Para ello antes hemos de añadir el ROLE\_USER de acceso a todos los usuarios.

En el modelo de user hemos de tener esto:

public function getRoles(){

        return array('ROLE\_USER');

}

Ahora simplemente ponemos las rutas que queremos que únicamente sean accedidas por usuarios logueados.

- { path: ^/tasks, role: ROLE\_USER }

- { path: ^/create, role: ROLE\_USER }

- { path: ^/edit/, role: ROLE\_USER }

- { path: ^/delete/, role: ROLE\_USER }

Y con esto, queda finalizado nuestro proyecto.

# Dependencias

Dependencias de 01-Laravel:

* Xampp con PHP 8
* Composer 2.0.7
* Laravel 8
* MySQL

Dependencias de 02-Symfony:

* Xampp con PHP 8
* Composer 2.0.7
* Symfony 5
* MySQL