**Trash Recolector**

**documentación de codigo**

Introducción

#### Descripción Breve del Problema:

La subsecretaría de medio ambiente del municipio de Tandil nos contactó para que los ayudemos con el desarrollo de una aplicación que permita generar un mapa de la basura en la ciudad. Se necesita contar con un mecanismo que permita a los ciudadanos reportar lugares de la ciudad en los que se encuentre basura.

Herramientas utilizadas

Como primera instancia se definió y debatió, en base a los conocimientos en común de los integrantes, con que tecnología/s se iba a desarrollar el trabajo propuesto, y las pautas generales de desarrollo, para así todos estar trabajando de la misma manera y no tener problemas a la hora de lectura, fix de bugs, extensión de funcionalidades, etc., del código echo por cualquier integrante del equipo

En base a los conocimientos en común se decidió desarrollar la aplicación con la propuesta de diseño MVC (model-view-controller) y realizar una vista de la aplicación “movile-only”

#### Lenguajes de programación, de marcado y frameworks utilizados

* PHP
* JavaScript
* HTML 5
* CSS3
* Smarty
* MySQL

#### Otros Softwares de utilidad

* Xampp (se decidió mostrar una versión local de la aplicación)
* Editores de texto varios (Atom, VScode, NotePad ++, etc.)

*Clases Intervinientes en el Desarrollo del Sistema*

Clases Correspondientes al Model:

#### Form Model

La funcionalidad principal de esta clase es de guardar los datos provistos por el formulario principal de la aplicación (denuncia de ciudadano).

#### Funciones que intervienen en la clase

**createAComplaint**: Dado los datos provisto por el formulario la función se encarga de guardar dichos datos en la Base de datos de forma segura y utilizando buenas practicas

**uploadImage**: Dada una imagen provista por el formulario, esta función se encarga de guardarla en la base de datos

**getUserComplaint**: Esta función se encarga de traer todas las denuncias realizadas por un usuario en particular provisto por el sistema

#### Admin Model

La funcionalidad principal de esta clase traer los datos de las denuncias para luego poder ser utilizadas en la sesión de administrador

#### Funciones que intervienen en la clase

**fetchMap**: Se encarga de traer las coordenadas de los registros de denuncia guardados

**fetchAllReports**: Retorna todos los reportes que contenga la base de datos, independientemente del estado de dichos reportes

**changeStatus**: Se encarga de cambiar el estado de una denuncia en particular

#### Login Model

La funcionalidad principal de esta clase traer los datos de los usuarios registrados en la aplicación

#### Funciones que intervienen en la clase

**getUsers**: Retorna de la base de datos todos los usuarios guardados que contenga la aplicación

**getUser**: Retorna un usuario en particular

**insertUser**: Guarda un nuevo usuario en la base de datos

Clases Correspondientes a la View:

#### Form View

Se encarga de la vista del formulario de denuncia, para mostrar la información del lado del cliente se utiliza el framework *Smarty*

#### Funciones que intervienen en la clase

**showForm**: Muestra el template asociado al formulario de denuncia principal

**successfulComplaint**: Dada una denuncia, en caso de ser exitosa se mostrara un template para dar seguridad al usuario de que su denuncia fue tomada

**showVisitorPage**: Muestra el template con la vista que recibe un visitante no registrado en la aplicación

**showAdminPage**: Muestra el template con la vista asociada a la página de administrador, con todas las funcionalidades que propias del administrador de la aplicación

**showUserPage**: Muestra el template con la vista asociada a la página de usuario proveyendo de todas las funcionalidades requeridas para un usuario

#### Admin View

Se encarga de la vista asociada al administrador, para esto es utilizado el framework *Smarty*

#### Funciones que intervienen en la clase

**showMap**: Muestra el mapa con la ruta de la basura para que el Administrador decida el mejor camino

**showReports**: Muestra todas las denuncias de los usuarios con sus respectivos detalles

#### Login View

Se encarga de la vista asociada al login y el registro, para esto es utilizado el framework *Smarty*

#### Funciones que intervienen en la clase

**showLogin**: Muestra la pantalla principal con un formulario de loguea para que así el usuario pueda entrar a la aplicación

**showRegistrationForm**: Muestra el formulario de registro para un nuevo usuario

Clases Correspondientes al Controller:

#### Form Controller

Se encarga de conectar la view y el model correspondientes al formulario de denuncia

#### Funciones que intervienen en la clase

**home**: Llama a la view y pide que muestre el formulario principal de denuncia

**successfulComplaint**: Se encarga de pedirle a la view que informe de cuando una denuncia es exitosa

**addComplaint:** Dado los datos ingresados en el formulario de denuncia, esta función se encarga de pasárselos al model para que luego sean guardados en la Base de Datos

#### Admin Controller

Se encarga de conectar la view y el model correspondientes al área de administrador

#### Funciones que intervienen en la clase

**map**: Le facilita los datos a la view para que así pueda mostrar en el mapa los lugares donde existen denuncias

**report**: Pide los reportes de basura al Model y se los pasa a la View para ser visualizados en el área de administrador

**editarEstado**: Coordina al model para que cambie el estado de una denuncia en particular

#### Login Controller

Se encarga de conectar la view y el model correspondientes al área de login y registro

#### Funciones que intervienen en la clase

**login**: Informa a la View que tiene que mostrar el template asociado a la página de login (página principal)

**logout**: Se encarga de dar por terminada la sesión del usuario logueado que la solicite

**createAcount**: Informa a la View para que muestre el template asociado al registro de un usuario

**verifyRegister**: Se encarga de registrar un usuario en caso de que el registro sea exitoso, loguea automáticamente al nuevo usuario en la aplicación

**verifyLogin**: Verifica que el usuario que está solicitando un “log in” exista en la base de datos y que sus credenciales sean correctas, en caso de ser así da como iniciada la sesión del usuario a la aplicación