



INSTITUTO TECNOLÓGICO DE TLAXIACO



ING. EN SISTEMAS COMPUTACIONALES

MATERIA:

INGENIERIA DE SOFTWARE

Practica #5:

DOCUMENTO DE ARQUITECTURA

DOCENTE:

JOSÉ ALFREDO ROMAN CRUZ.

ALUMNO:

ELVIS CRUZ BAUTISTA

SAMUEL JOSÉ ANTONIO

ANA LAURA APARICIO HERNÁNDEZ

MARÍA CONCEPCIÓN GARCÍA CASTRO

GRUPO

7US SEMESTRE

20 DE JUNIO DE 2021 TLAXIACO, OAXACA

Contenido

Tabla de ilustración	3
INTRODUCCIÓN.....	5
INTEGRANTES DEL PROYECTO.....	6
ALCANSE DE LA ARQUITECTURA.....	6
OBJETIVOS DE LA ARQUITECTURA	7
INTERFAZ GRAFICA DE LA APLICACIÓN.....	8
Prototipo (Mockups) en ADOBE XD.....	8
Actualización de la aplicación.....	15
DIAGRAMA DE CLASES.....	18
DIAGRAMA DE SECUENCIA	19
Vista de Datos (Diagrama Relacional).....	20
DIAGRAMA DE COMPONENTES	21
DIAGRAMA DE ESTADOS.....	21
DIAGRAMA DE DESPLIEGUE	22
MODELO VISTA CONTROLADOR.....	23
Aplicación de la metodología SCRUM para el desarrollo	26
Beneficios	27
Sprint #1.....	28
Front-End.....	28
Back-End	29
Sprint #2.....	35
Front-End	35
Back-End	36
Diseño de la aplicación final	37
Codificación de Front-End.....	40
Codificación Back-End.....	41
Resultado final	43
Conclusión	46
REFERENCIA.....	47

Tabla de ilustración

ILUSTRACIÓN 1 VISTA PRINCIPAL DE LA APLICACIÓN	8
ILUSTRACIÓN 2 LOGEO	9
ILUSTRACIÓN 3 REGISTRO DE LOCAL	10
ILUSTRACIÓN 4 REGISTRO DE USUARIO	11
ILUSTRACIÓN 5 VISTA DEL LOCAL	12
ILUSTRACIÓN 6 HORARIOS DE SERVICIO.....	13
ILUSTRACIÓN 7 VISTA FINAL.....	14
ILUSTRACIÓN 8 ACTUALIZACIÓN DE LOGIN.....	15
ILUSTRACIÓN 9 ACTUALIZACIÓN DE REGISTRO.....	15
ILUSTRACIÓN 10 CREAMOS LA VISTA HOME.....	16
ILUSTRACIÓN 11 CREAMOS LA VISTA HISTORIAL.....	16
ILUSTRACIÓN 12 CREAMOS LA VISTA MAPA.....	16
ILUSTRACIÓN 13 CREAMOS LA OPCIÓN DE PERSONALIZAR USUARIO.....	17
ILUSTRACIÓN 14 CREAMOS OPCIÓN DE CONTRASEÑA.....	17
ILUSTRACIÓN 15 DIAGRAMA DE CLASES	18
ILUSTRACIÓN 16 DIAGRAMA DE SECUENCIA.....	19
ILUSTRACIÓN 17 VISTA DE DATOS	20
ILUSTRACIÓN 18 DIAGRAMA DE COMPONENTES.....	21
ILUSTRACIÓN 19 DIAGRAMA DE ESTADO	21
ILUSTRACIÓN 20 DIAGRAMA DE DESPLIEGUE	22
ILUSTRACIÓN 21 MVC	25
ILUSTRACIÓN 22 EN PROCESO (1-2-3) DE DESARROLLO	28
ILUSTRACIÓN 23 EN PROCESO (4-5-6) DE DESARROLLO	28
ILUSTRACIÓN 24 EN PROCESO (7-8-9) DE DESARROLLO	28
ILUSTRACIÓN 25 EN PROCESO (1-2-3-4) DE DESARROLLO	29
ILUSTRACIÓN 26 EN PROCESO (3-4-5) DE DESARROLLO	29
ILUSTRACIÓN 27 CARPETAS DE MVC	30
ILUSTRACIÓN 28 ARCHIVO CONTROLLER.PHP.....	30
ILUSTRACIÓN 29 ARCHIVO MODELO.PHP.....	31
ILUSTRACIÓN 30 ARCHIVO DE VISTA.PHP.....	32
ILUSTRACIÓN 31 ABRIMOS APLICACION Y HACEMOS LOGEO	33
ILUSTRACIÓN 32 LOGEO USUARIO	33
ILUSTRACIÓN 33 REGISTRO DE NUEVO LOCAL	33
ILUSTRACIÓN 34 USUARIO Y NEGOCIO INSERTADOS.....	34

ILUSTRACIÓN 35 LOGEO DE USUARIO.....	34
ILUSTRACIÓN 36 BASE DE DATOS EN LINEA.	34
ILUSTRACIÓN 37 RETOMAMOS EL DESARROLLO (1-2-3).....	35
ILUSTRACIÓN 38 RETOMAMOS EL DESARROLLO (4-5-6).....	35
ILUSTRACIÓN 39 RETOMAMOS EL DESARROLLO (7-8-9).....	35
ILUSTRACIÓN 40 RETOMAMOS EL DESARROLLO (1-2-3-4)	36
ILUSTRACIÓN 41 RETOMAMOS EL DESARROLLO DE (3-4-5)	36
ILUSTRACIÓN 42 INICIO DE LA APLICACIÓN.....	37
ILUSTRACIÓN 43 LOGIN.....	37
ILUSTRACIÓN 44 VISTA DE REGISTRO.....	37
ILUSTRACIÓN 45 VISTA HOME	38
ILUSTRACIÓN 46 SECCIÓN DE HISTORIAL.....	38
ILUSTRACIÓN 47 MAPA DE LA APLICACIÓN.	38
ILUSTRACIÓN 48 USUARIO Y CONTRASEÑA	39
ILUSTRACIÓN 49 ACTUALIZAR CONTRASEÑA.....	39
ILUSTRACIÓN 50 PÁGINA PRINCIPAL.....	40
ILUSTRACIÓN 51 PÁGINA DE LOGIN.....	40
ILUSTRACIÓN 52 PAGINA DE REGISTRO.	40
ILUSTRACIÓN 53 CONTROLADOR PRINCIPAL PARA LA VISTA HOME	41
ILUSTRACIÓN 54 CONEXIÓN A LA BASE DE DATOS.....	41
ILUSTRACIÓN 55 CONTROLADOR DE HISTORIAL.....	41
ILUSTRACIÓN 56 ARCHIVO DE AUTENTIFICACIÓN DE USUARIOS.....	42
ILUSTRACIÓN 57 ARCHIVO Js.....	42
ILUSTRACIÓN 58 ARCHIVOS PARA BUGS.....	43
ILUSTRACIÓN 59 VISTA HOME	43
ILUSTRACIÓN 60 SECCIÓN DE HISTORIAL	44
ILUSTRACIÓN 61 SECCIÓN MAPA	44
ILUSTRACIÓN 62 NOMBRE Y FOTO CARGADOS EXITOSAMENTE.....	45
ILUSTRACIÓN 63 PANEL DE ACTUALIZAR LA CONTRASEÑA.....	45

INTRODUCCIÓN

Mediante este documentos se pretende realizar un informe sobre todos los puntos generales con las cuales se aplicaran en la aplicación y la forma en la que estará estructurara o bien la arquitectura de la aplicación como el prototipo, diseño (mockups) y la actualización de este mismo, la base de datos ya no se le realizara ninguna modificación ya que la estructura ya está definida, codificación, etc. De igual manera vamos a comprender la estructura del **MODELO VISTA CONTROLADOR** o bien **MVC** con la cual vamos a trabajar para estructurar nuestra aplicación y de igual manera conocer que es MVC y para que nos sirve este modelo.

También se documentara la codificación del Front-End y el Back-End del desarrollo de la aplicación que será desarrollado mediante arquitectura MVC y realizar pruebas para que la aplicación sea funcional.

INTEGRANTES DEL PROYECTO

ROLES	NOMBRES
Líder del proyecto	SAMUEL JOSE ANTONIO
Programador y Analista	ELVIS CRUZ BAUTISTA
Diseñadora	ANA LAURA APARICIO HERNANDEZ
Tester	MARIA CONCEPCIÓN

ALCANSE DE LA ARQUITECTURA.

La arquitectura de software tiene un rol muy importante en el desarrollo de software ya que mediante este nos basaremos para un buen desarrollo de la aplicación aplicando las funcionalidades correctamente como puede ser requisitos funcionales o no funcionales para mostrar una buena calidad de software al usuario y que el sistema sea seguro, confiable y que tenga un buen rendimiento.

El sistema que pretendemos realizar será una aplicación hibrida ya que deseamos que se pueda acceder desde una plataforma web, como de igual manera desde un dispositivo móvil. Pero por el momento solo se desarrollara en la plataforma web. Para poder utilizar la aplicación web y tener acceso a nuestra plataforma solo los que deseen registrar o agregar algún local con el que cuenten solo serán los que podrán crear un registro en el cual pondrán los datos del usuario, nombre del local y la ubicación para que este puede visualizarse en el mapa con el que contaremos. Recalcando el software será de aplicación web, por lo que se ejecutara mediante una computadora accediendo desde nuestra página web.

OBJETIVOS DE LA ARQUITECTURA

Funcionalidad: Buen diseño tal que las funcionalidades asignadas tengan un buen funcionamiento al momento de interactuar con este.

Mantenible: Que tenga buen funcionamiento y a su vez que se pueda seguir actualizando para un mejor funcionamiento del software.

Confiabilidad: No debe causar daños físicos o económicos en el caso de fallos.

Seguridad: El software contara con seguridad mediante encriptación de contraseña la cual estará basada mediante h5 la cual nos ofrece un servicio de encriptación para proteger a los usuarios de ataques a su cuenta o que la contraseña sea más segura si se desea extraer desde la base de datos, razón por la que se eligió el tipo de encriptación.

Eficiencia: El sistema no debe desperdiciar los recursos del sistema.

Expansibilidad: Mantendrá la expansibilidad necesaria para facilitar ese proceso de desarrollar versiones posteriores

Portabilidad: El sistema se desarrollara mediante PHP, MYSQL Y JAVASCRIPT, ya que al crear la página mediante estos componentes será un sistema más confiable y con mejor funcionalidad.

Reutilización: El sistema puede transferirse de un sistema a otro.

INTERFAZ GRAFICA DE LA APLICACIÓN

Prototipo (Mockups) en ADOBE XD

En la ventana principal de nuestra aplicación se encontraran las ubicaciones de los locales registrados desde nuestra aplicación.

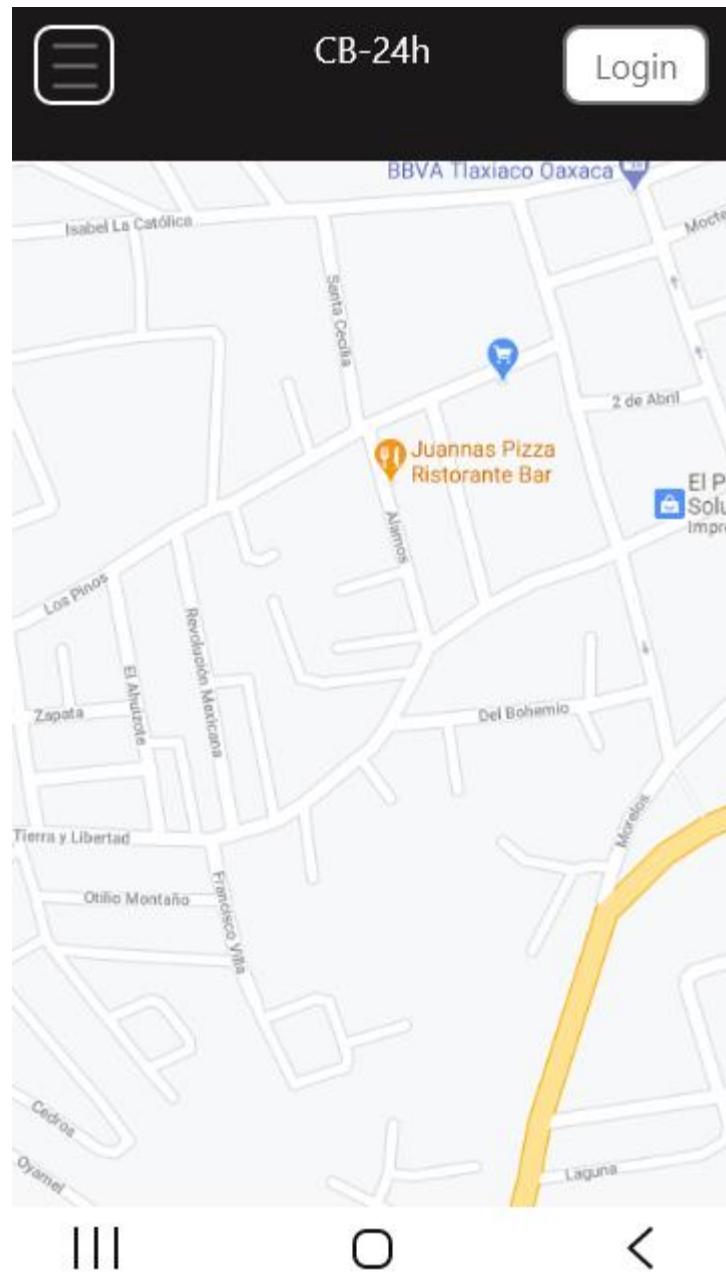


Ilustración 1 Vista principal de la aplicación.

El usuario podrá logearse mediante un usuario que el haya creado o bien acceder desde las opciones que se ofrecen en nuestra aplicación mediante FACEBOOK o GMAIL.



Ilustración 2 Logeo.

Si el usuario cuenta con algún local y desea darse a conocer entonces tendremos la opción de REGISTRAR LOCAL en la cual describirá ubicación, descripción, nombre de usuario con la cual le permitirá poder fijar ubicación y más detalles del local. De igual manera se cuenta con la opción de un usuario normal, la cual es el consumidor.

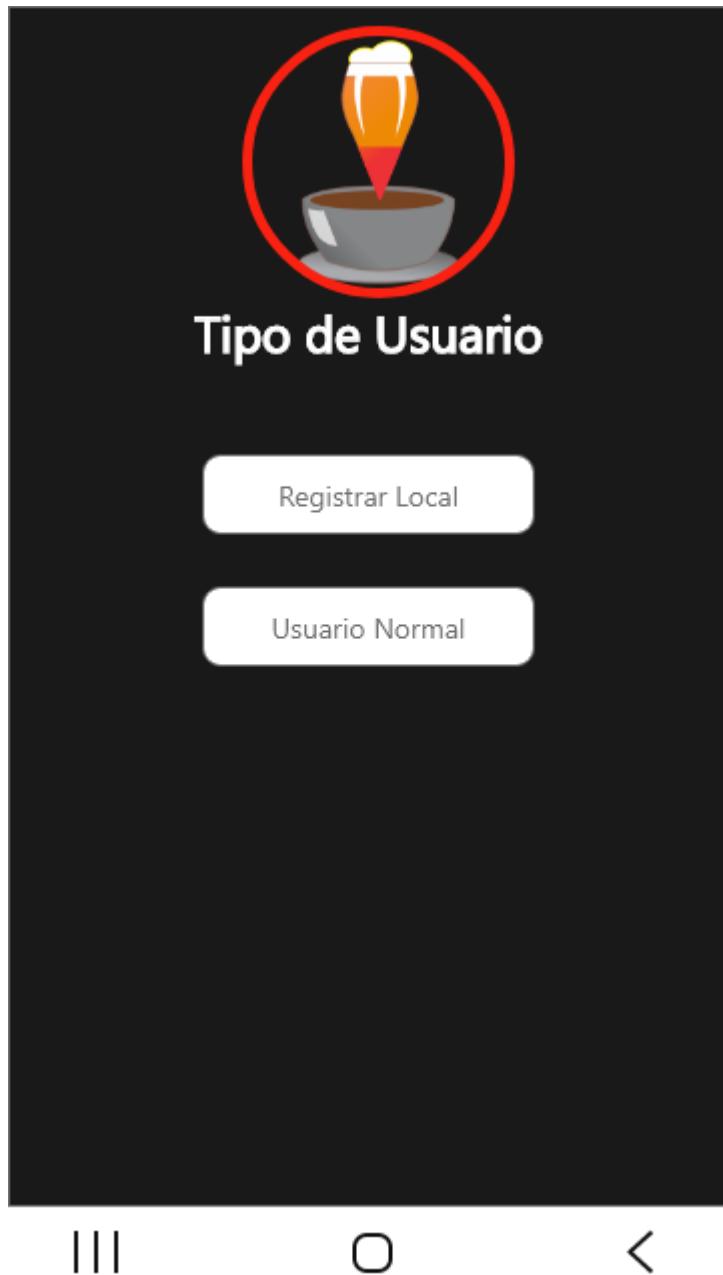


Ilustración 3 Registro de local.

Si el usuario cuenta con local entonces contaremos con la siguiente ventana donde el usuario que desea dar a conocer el local pondrá los datos que se le pide.



Ilustración 4 Registro de usuario.

Una vez que el usuario se haya registrado y logeado tendremos la siguiente ventana donde tendrá las opciones de mostrar detalladamente el local dando referencias de su local.



Ilustración 5 Vista del local.

Por último el usuario del local fijara los horarios en el que abren el local, tendrá la opción de poner si cuenta con servicio a domicilio o no, y los días que está abierto el local.

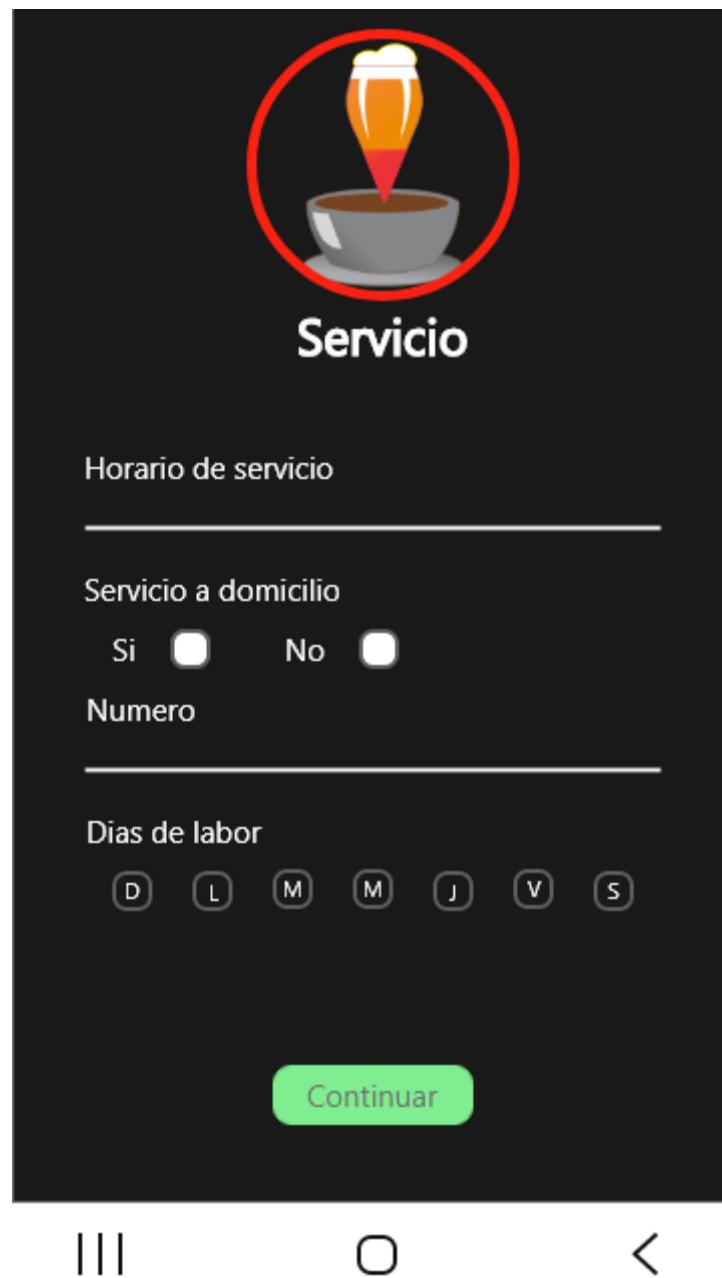


Ilustración 6 Horarios de servicio.

Como último punto es como se muestra al usuario normal al momento de buscar algún local fijado o registrado desde nuestra aplicación donde el usuario tendrá la opción de ver los horarios, el menú del local, los días y una breve reseña del local.

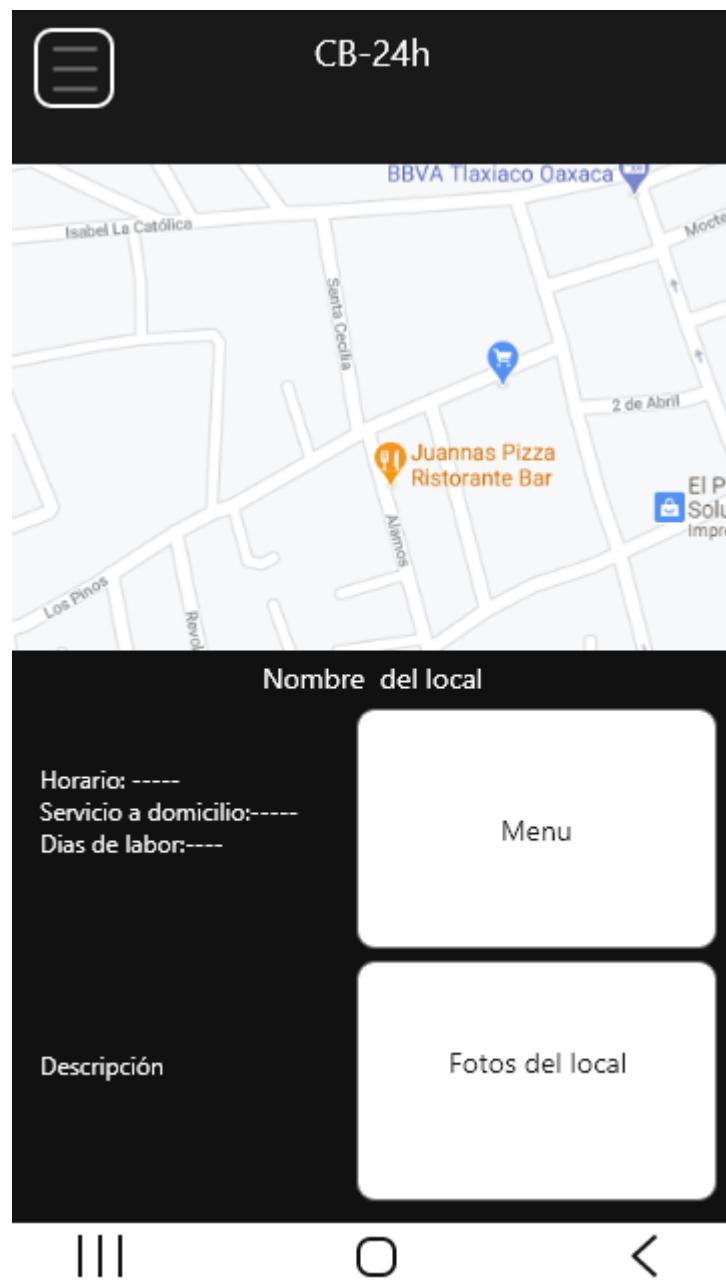


Ilustración 7 Vista final.

Actualización de la aplicación

Conforme a los primeros diseños que se habían realizado para la aplicación y en su primera etapa de desarrollo se ha decidido actualizar nuestra aplicación para esta segunda etapa de desarrollo.

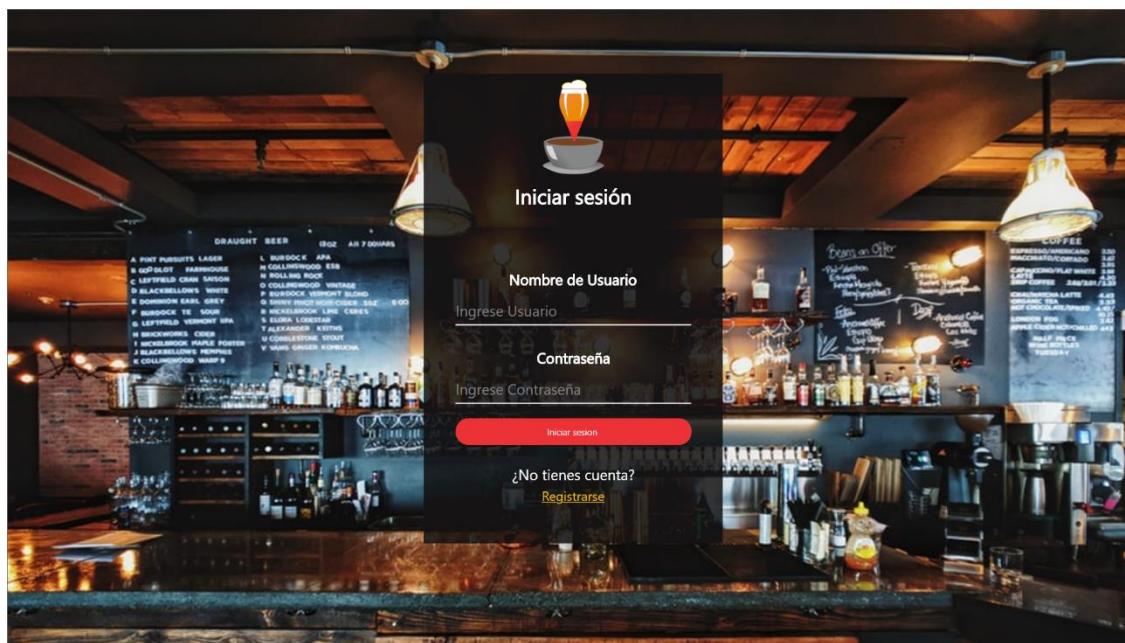


Ilustración 8 Actualización de Login.

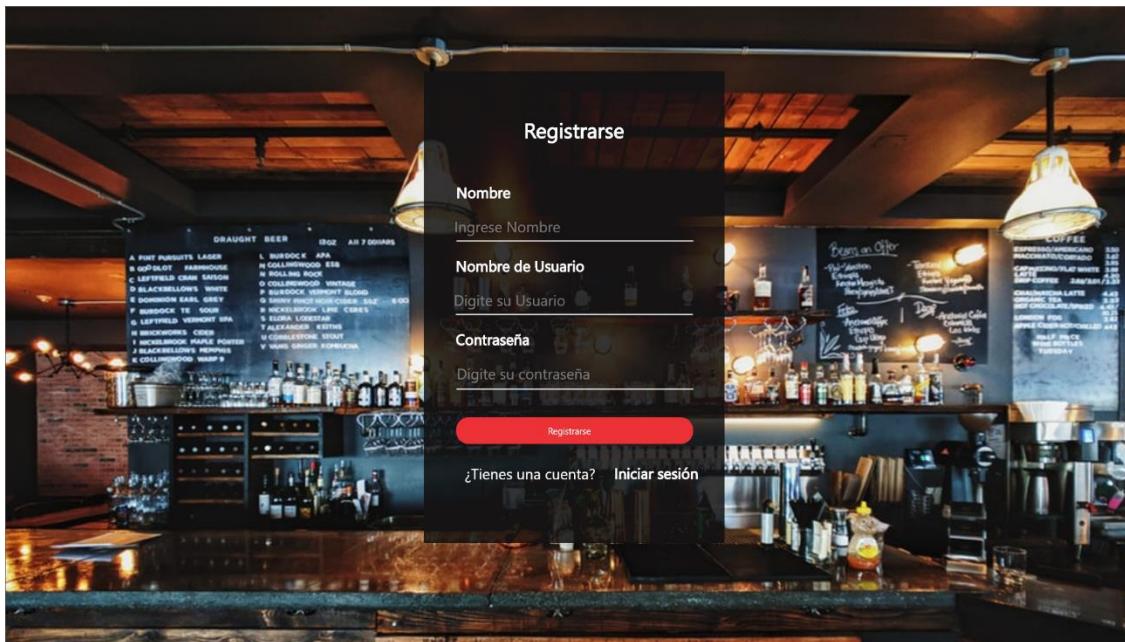


Ilustración 9 Actualización de Registro.

De igual manera se actualizo la vista de Home para cuando el usuario acceda a su cuenta.

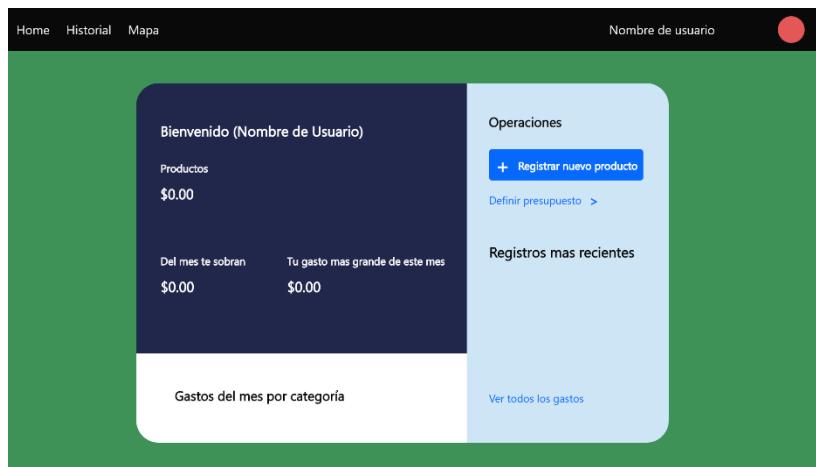


Ilustración 10 creamos la vista **Home**.

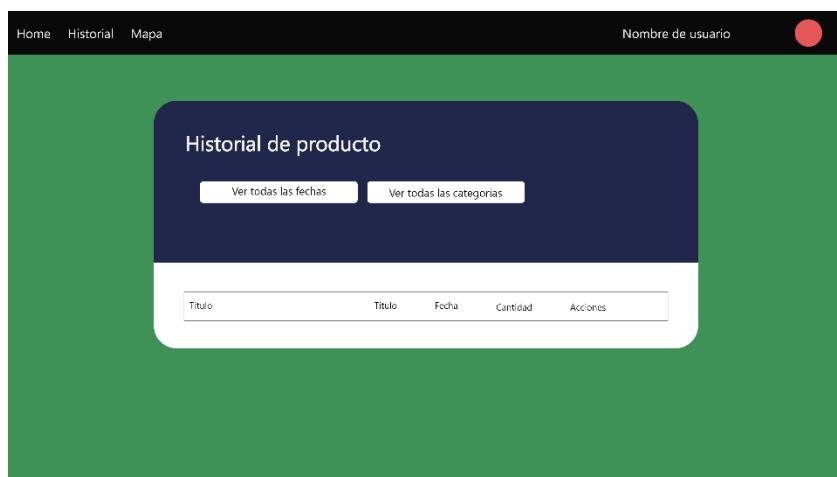


Ilustración 11 Creamos la vista **Historial**.

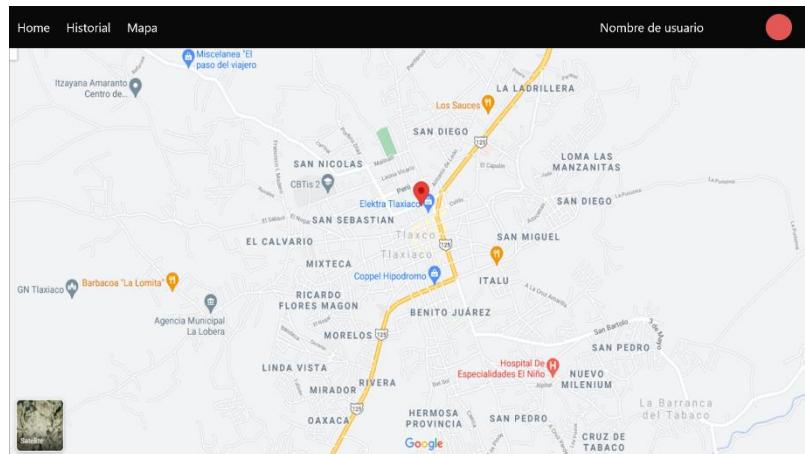


Ilustración 12 Creamos la vista **Mapa**.

Creamos una opción de perfil para que el usuario pueda modificar el nombre y asignar una foto para su perfil de usuario.

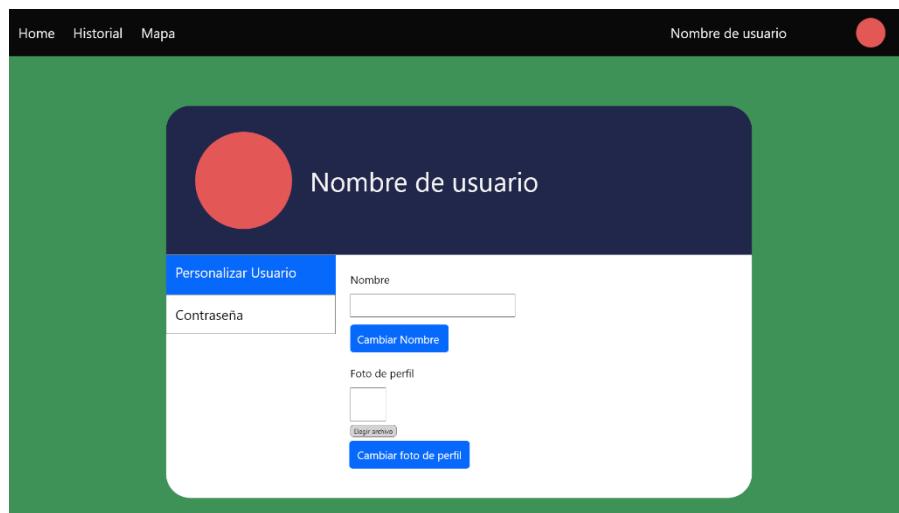


Ilustración 13 Creamos la opción de Personalizar usuario.

De igual manera creamos una opción para que el usuario pueda modificar su contraseña si es que así lo desea.

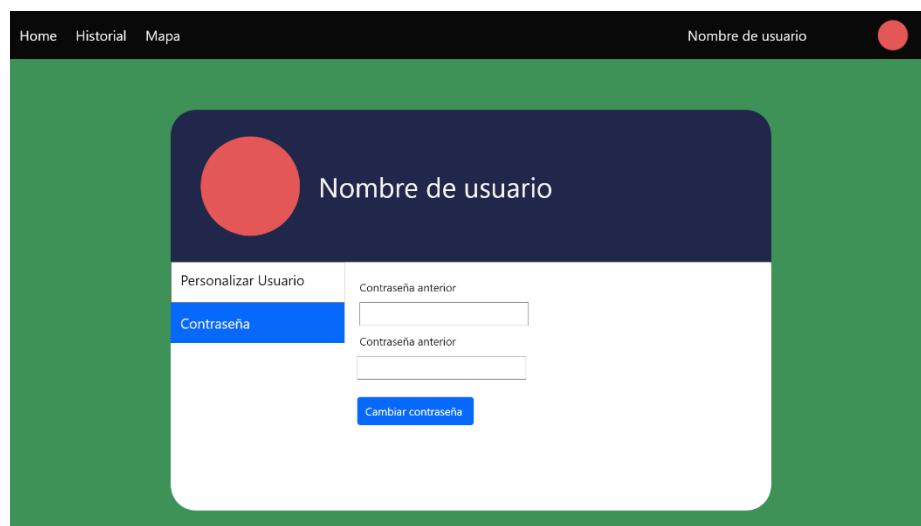


Ilustración 14 Creamos opción de contraseña.

DIAGRAMA DE CLASES

El diagrama de clase es un tipo de diagrama que nos permite realizar una estructura donde se describimos con los puntos más importantes con la cual contara nuestro software y la cual será base fundamental para el buen funcionamiento, para el desarrollo de nuestra aplicación el diagrama de clases queda de la siguiente manera.

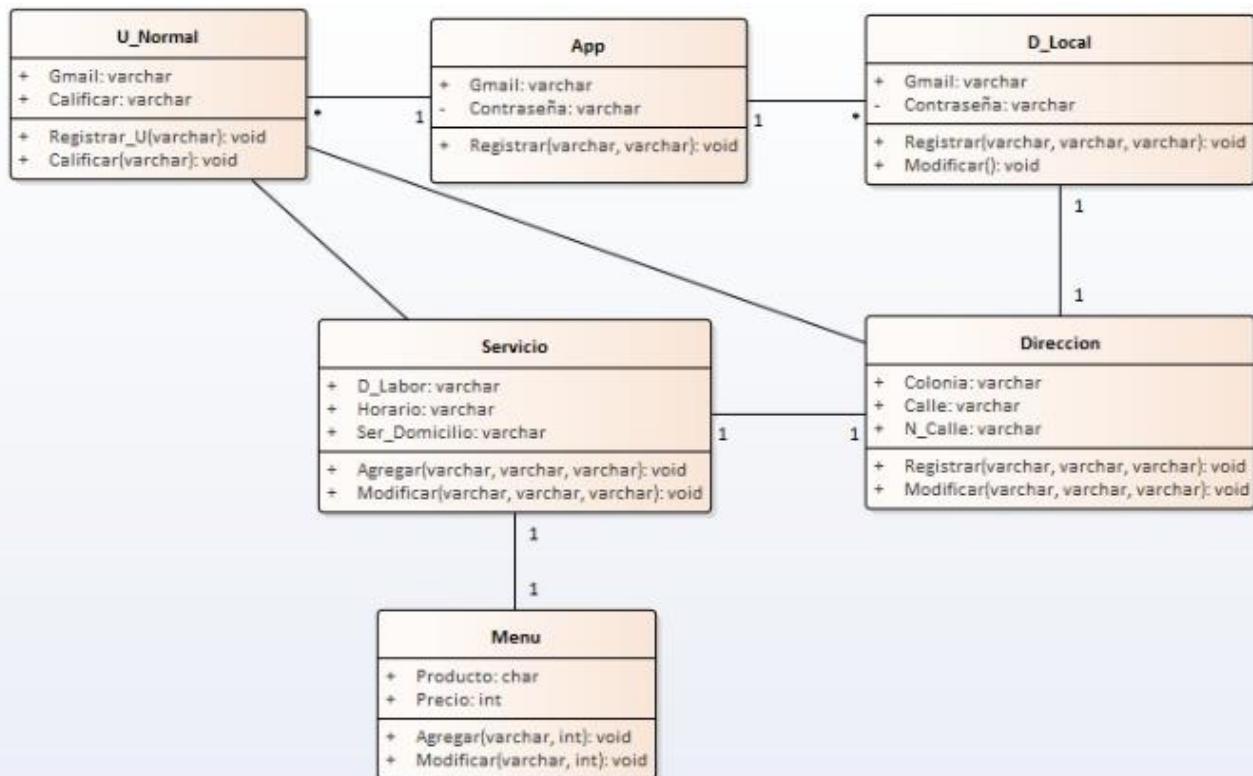


Ilustración 15 Diagrama de clases.

DIAGRAMA DE SECUENCIA

El diagrama de secuencia es un tipo de diagrama usado para modelar interacción entre objetos en un sistema según sea su funcionalidad de dicho sistema, en este caso el diagrama de secuencias del proyecto es el siguiente:

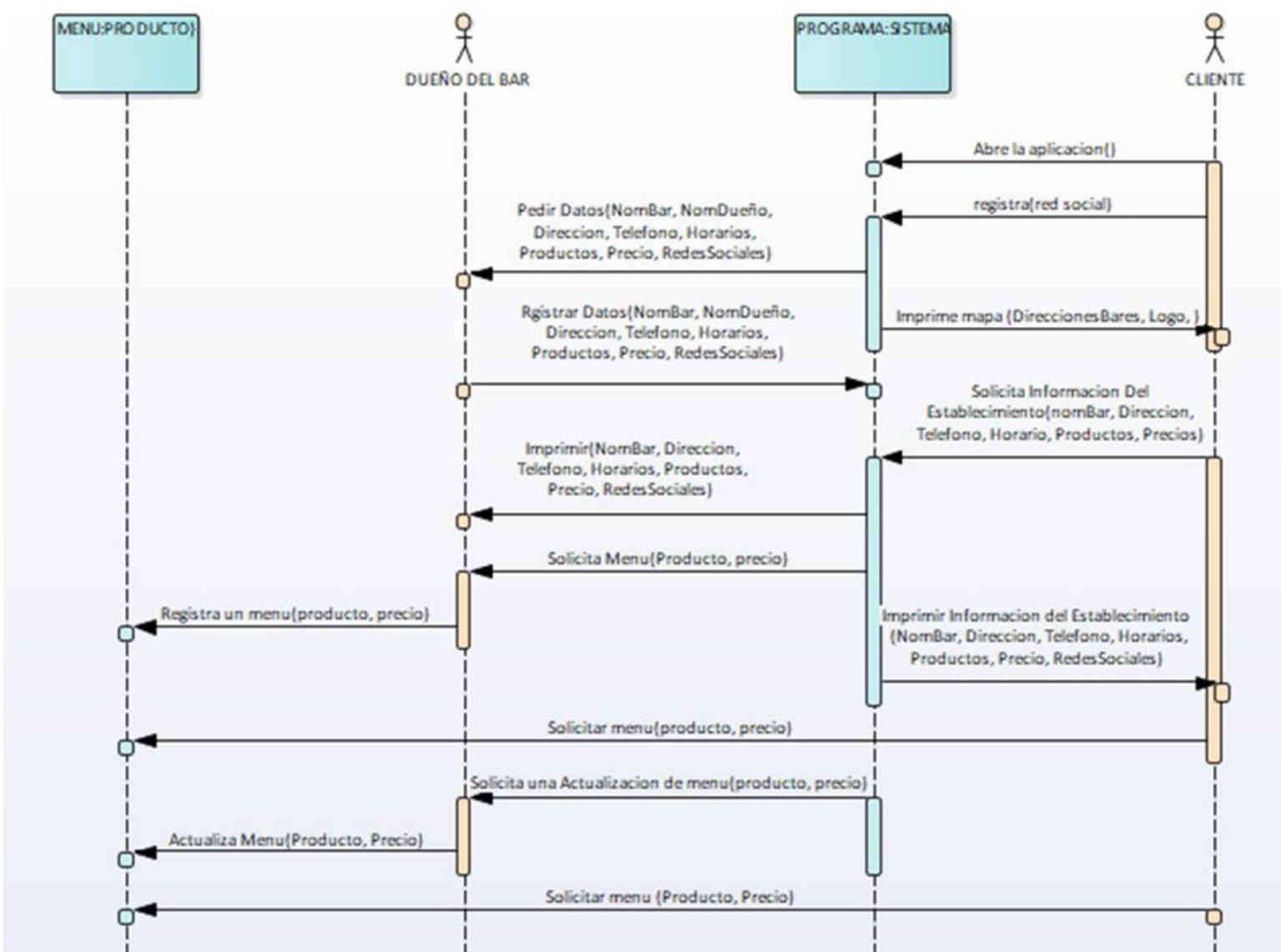


Ilustración 16 Diagrama de secuencia.

Vista de Datos (Diagrama Relacional)

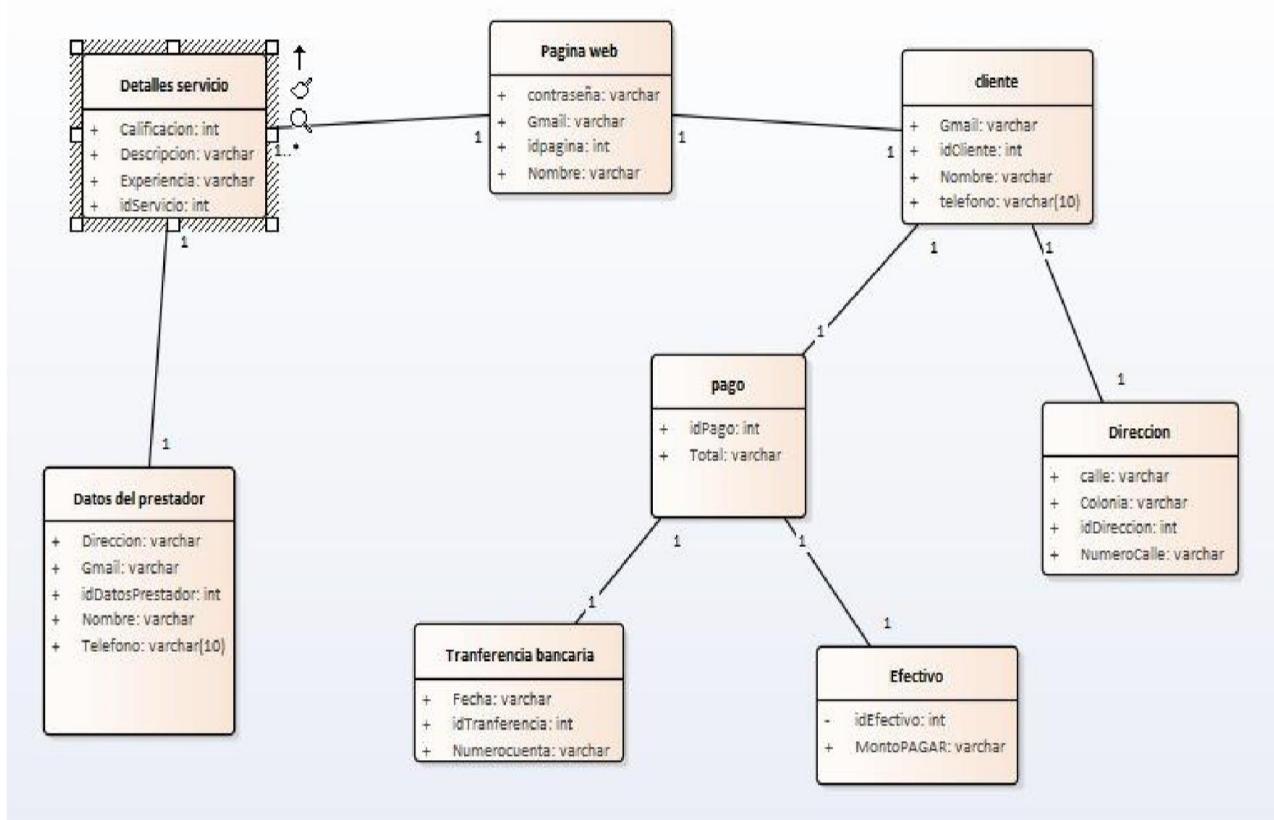


Ilustración 17 Vista de datos.

DIAGRAMA DE COMPONENTES

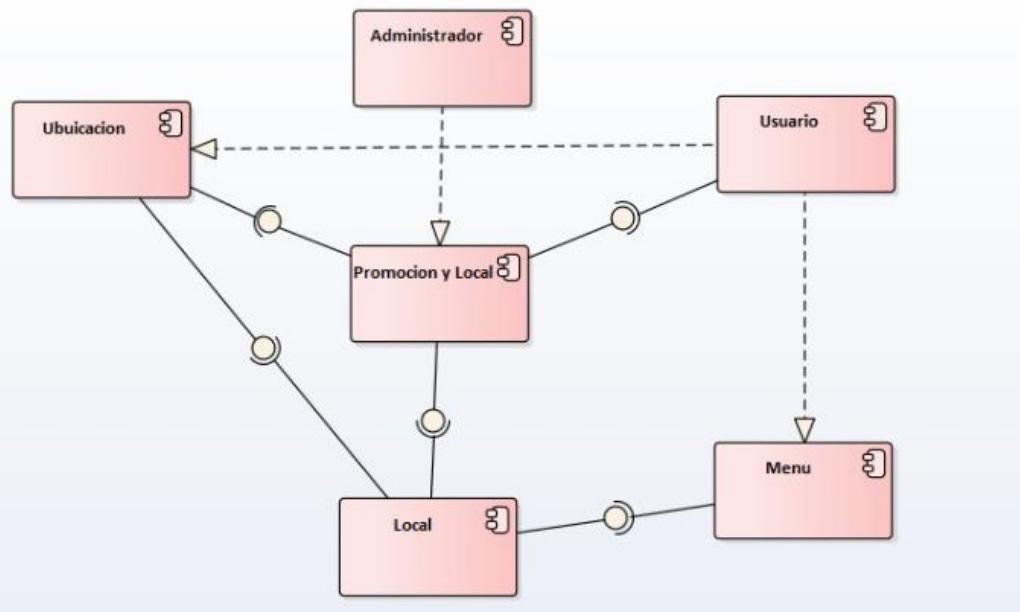


Ilustración 18 Diagrama de componentes.

DIAGRAMA DE ESTADOS

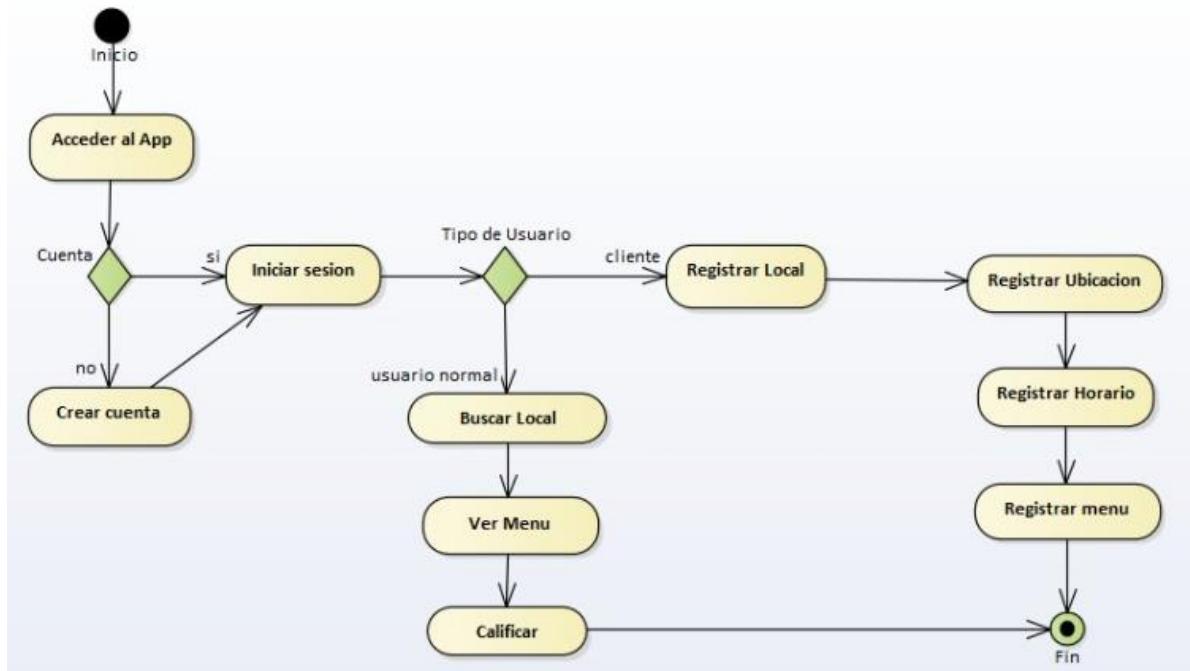


Ilustración 19 Diagrama de estado.

DIAGRAMA DE DESPLIEGUE

Los diagramas de despliegue se utilizan para visualizar los procesadores nodos/dispositivos de hardware de un sistema, los enlaces de comunicación entre ellos y la colocación de los archivos de software en ese hardware.

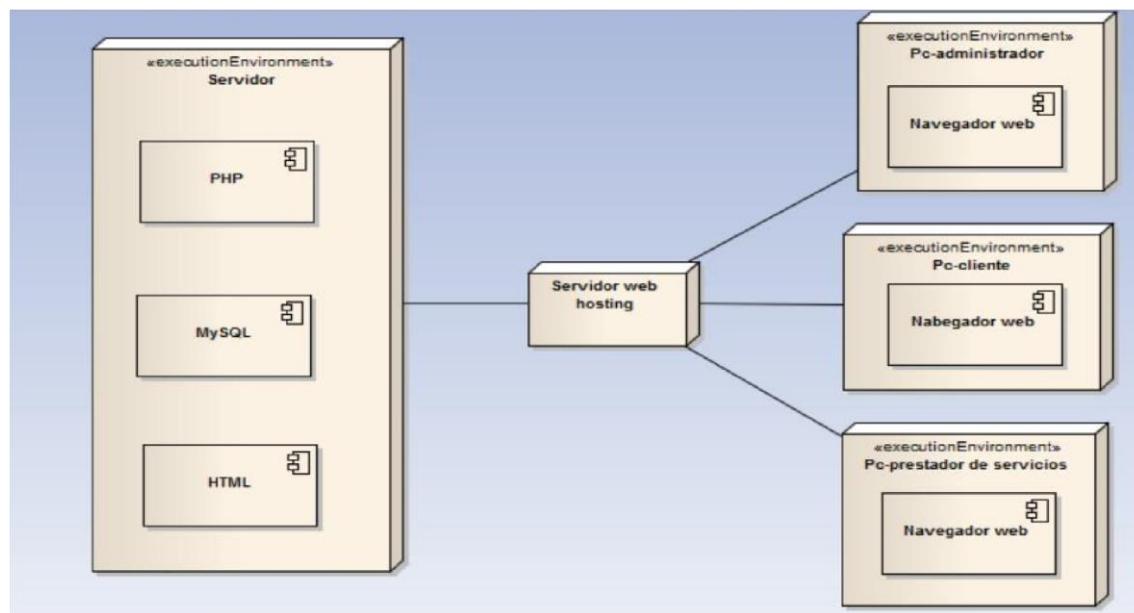


Ilustración 20 diagrama de despliegue

MODELO VISTA CONTROLADOR

Es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

- ❖ El **Modelo** que contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.
- ❖ La **Vista**, o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos interacción con éste.
- ❖ El **Controlador**, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

El modelo es el responsable de:

- ❖ Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- ❖ Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".
- ❖ Lleva un registro de las vistas y controladores del sistema.
- ❖ Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero por lotes que actualiza los datos, un temporizador que desencadena una inserción, etc.).

El controlador es responsable de:

- ❖ Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
- ❖ Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar ()". Una petición al modelo puede ser "Obtener_tiempo_de_entrega (nueva_orden_de_venta)".

Las vistas son responsables de:

- ❖ Recibir datos del modelo y los muestra al usuario.
- ❖ Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
- ❖ Pueden dar el servicio de "Actualización()", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

Capa de Interfaz Gráfica: Es la capa que presenta todo lo que el usuario puede observar, interactuar, utilizar, consultar datos del sistema. Dependiendo el tipo de usuario, la interacción con el sistema se restringe y la interfaz cambia.

Capa de Lógica: En esta capa se reciben las llamadas al sistema para realizar la conexión entre la interfaz gráfica y la base de datos.

Capa de Persistencia: Esta permite las conexiones de la base de datos con la interfaz gráfica, es la que ejecutan el comando de llamas al sistema.

INTERFAZ

Autenticación de usuario, se exigen dos requerimientos de información para ingresar al sistema correo electrónico y contraseña. Ingresar usuario y contraseña, de no estar registrado acceder a registro crear cuenta

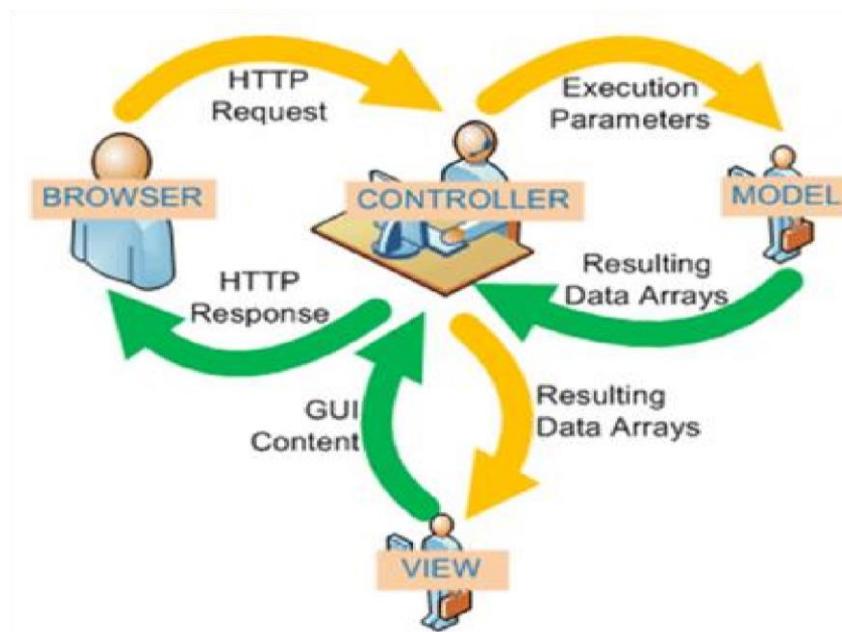


Ilustración 21 MVC.

Aplicación de la metodología SCRUM para el desarrollo

Al implementar la metodología SCRUM se tiene como objetivo desarrollar la aplicación de una manera ágil generando entregas incrementales en lugar de realizar una planificación y ejecución completa del desarrollo del software.

Para el desarrollo de la aplicación web que permitirá el cálculo automático de horas extra para los conductores de una empresa de transporte se definió la metodología ágil scrum, la cual acelerará las entregas del proyecto para así suplir de manera rápida la necesidad que tiene esta empresa.

Estructura

El documento se divide en una serie de capítulos que a su vez se subdividen en apartados.

Los

Capítulos son los siguientes:

1. Introducción: da una visión global del proyecto, exponiendo los objetivos a alcanzar, el porque de la realización de este proyecto y el contexto.
2. Análisis del sistema: es un estudio previo al desarrollo en el cual se detalla las propiedades del sistema y los elementos que se deben desarrollar.
3. Desarrollo del sistema: en este capítulo se detalla todo lo relativo al propio desarrollo del proyecto.
4. Planificación y presupuesto: se expone la planificación previa basada en las estimaciones para la duración del proyecto y se realiza un pequeño análisis económico de éste para hallar el coste final de su realización.
5. Conclusiones: último capítulo del documento en el cual se plasman las conclusiones extraídas después de la finalización del proyecto, tanto personales como de resultados.

También se proponen desarrollos futuros que enriquezcan al sistema

Scrum es una metodología ágil y flexible para gestionar el desarrollo de software, cuyo principal objetivo es maximizar el retorno de la inversión para su empresa (ROI). Se basa en construir primero la funcionalidad de mayor valor para el cliente y en los principios de inspección continua, adaptación, auto-gestión e innovación.

Con la metodología Scrum el cliente se entusiasma y se compromete con el proyecto dado que lo ve crecer iteración a iteración. Asimismo le permite en cualquier momento realinear

el software con los objetivos de negocio de su empresa, ya que puede introducir cambios funcionales o de prioridad en el inicio de cada nueva iteración sin ningún problema.

Esta metódica de trabajo promueve la innovación, motivación y compromiso del equipo que forma parte del proyecto, por lo que los profesionales encuentran un ámbito propicio para desarrollar sus capacidades.

Beneficios

- **Cumplimiento de expectativas:** El cliente establece sus expectativas indicando el valor que le aporta cada requisito / **historia** del proyecto, el equipo los estima y con esta información el **Product Owner** establece su prioridad. De manera regular, en las demos de Sprint el **Product Owner** comprueba que efectivamente los requisitos se han cumplido y transmite feedback al equipo.
- **Flexibilidad a cambios:** Alta capacidad de reacción ante los cambios de requerimientos generados por necesidades del cliente o evoluciones del mercado. La metodología está diseñada para adaptarse a los cambios de requerimientos que conllevan los proyectos complejos.
- **Reducción del Time to Market:** El cliente puede empezar a utilizar las funcionalidades más importantes del proyecto antes de que esté finalizado por completo.
- **Mayor calidad del software:** La metódica de trabajo y la necesidad de obtener una versión funcional después de cada iteración, ayuda a la obtención de un software de calidad superior.
- **Mayor productividad:** Se consigue entre otras razones, gracias a la eliminación de la burocracia y a la motivación del equipo que proporciona el hecho de que sean autónomos para organizarse.
- **Maximiza el retorno de la inversión (ROI):** Producción de software únicamente con las prestaciones que aportan mayor valor de negocio gracias a la priorización por retorno de inversión.
- **Predicciones de tiempos:** Mediante esta metodología se conoce la velocidad media del equipo por sprint (los llamados puntos historia), con lo que consecuentemente, es posible estimar fácilmente para cuando se dispondrá de una determinada funcionalidad que todavía está en el Backlog.
- **Reducción de riesgos:** El hecho de llevar a cabo las funcionalidades de más valor en primer lugar y de conocer la velocidad con que el equipo avanza en el proyecto, permite despejar riesgos eficazmente de manera anticipada.

Sprint #1

En el sprint #1 hemos llevado un control sobre los tiempos de diseño y control de tiempo mediante un scrum.

Front-End

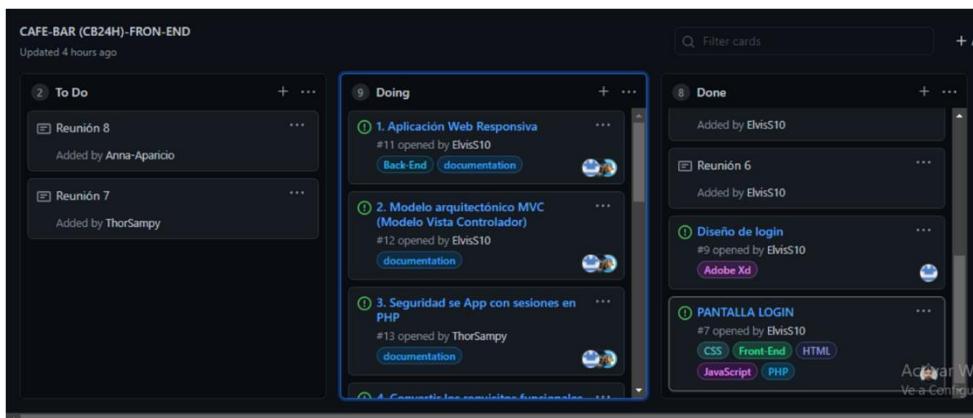


Ilustración 22 En proceso (1-2-3) de desarrollo.

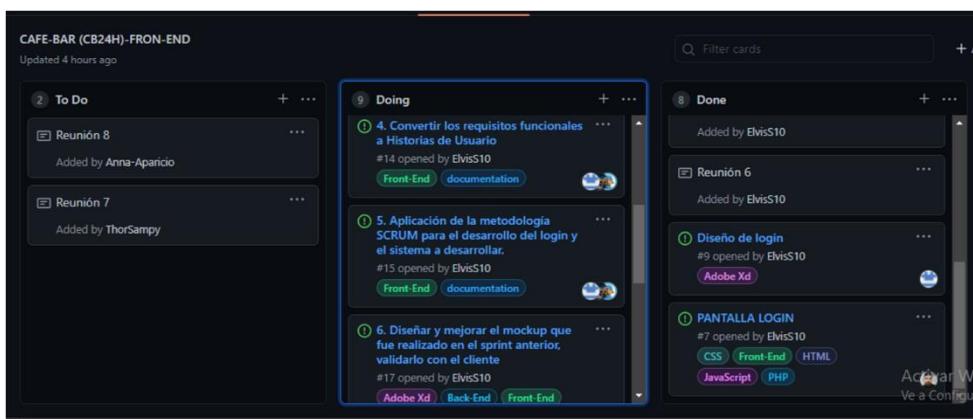


Ilustración 23 En proceso (4-5-6) de desarrollo.

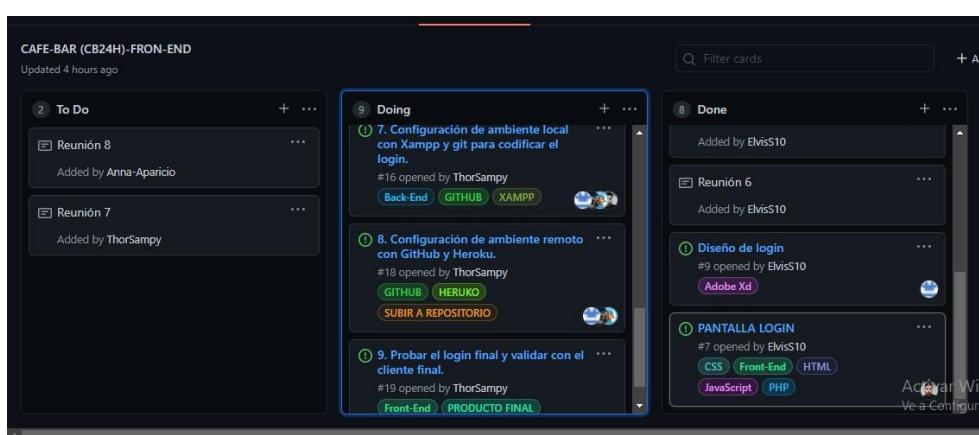


Ilustración 24 En proceso (7-8-9) de desarrollo

En el sprint #1 también contamos con un Scrum de los tiempos de desarrollo de la parte del Back-End, en la cual incluye reuniones que se realiza constantemente con el equipo de desarrollo.

Back-End

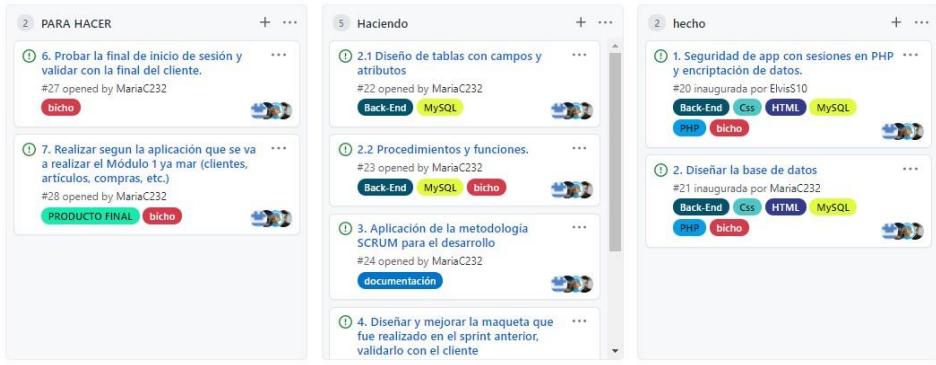


Ilustración 25 En proceso (1-2-3-4) de desarrollo

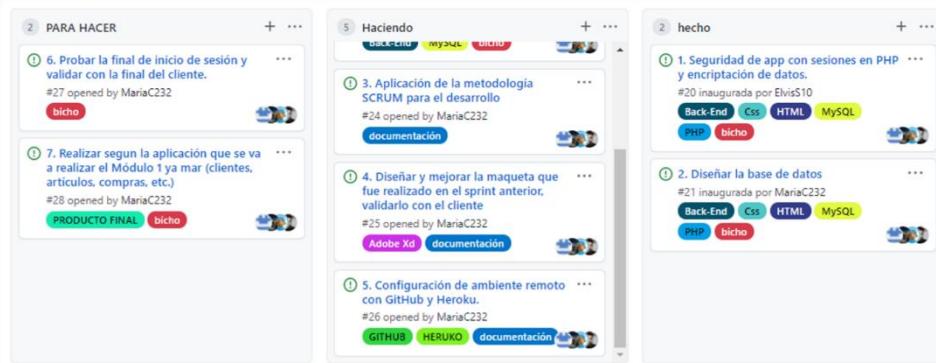


Ilustración 26 En proceso (3-4-5) de desarrollo.

Para crear nuestra aplicación vamos a utilizar y crear archivos que nos permitirá navegar entre los archivos y así cargar las rutas que definiremos para cada ventana que deseemos navegar, por lo que utilizaremos un modelo conocido como **MVC** (MODELO-VISTA-CONTROLADOR.).



Ilustración 27 Carpetas de MVC

A continuación mostramos la manera en la que se carga un controlador para luego poder visualizar nuestra vista.

```
<?php

class Main extends Controller
{
    function __construct()
    {
        parent::__construct();
        //echo "<p>Nuevo Main</P>";
    }
    function render()
    {
        $this->view->render('main/index');
    }

}
?>
```

Ilustración 28 Archivo controller.php.

En la parte de modelo es donde se ejecutara todo el código QUERY de MYSQL como la carga de datos del formulario en nuestra aplicación y es donde hará el proceso de inserción a la base de datos.

```
<?php

class RegistrarModel extends Model
{
    public function __construct()
    {
        parent::__construct();
    }

    public function insert($datos){
        //insertar datos en La BD Y PREPARAMOS EL QUERY DELA TABLA
        try
        {

//Preparamos la informacion antes de mandarla para evitar SQL inyección LINE 12
        $query = $this->db->connect()->prepare('INSERT INTO
CLIENTES(NOMBRE, USERNAME, SEGURIDAD, NEGOCIO, DESCRIPCION, UBICACION) VALUES
(:nombre, :username, :seguridad, :negocio, :descripcion, :ubicacion)');
        //mapeamos Los datos del arreglo LINE 13
        $query->execute(['nombre' => $datos['nombre'], 'username' => $datos[
'username'], 'seguridad' => $datos['seguridad'], 'negocio' => $datos['negocio'],
'descripcion' => $datos['descripcion'], 'ubicacion' => $datos['ubicacion']]);
        //echo "Dato Insertado";
        return true;

    } catch(PDOException $e)
    {
        //echo "Ya existe ese usuario";
        return false;
    }

}
}

?>
```

Ilustración 29 Archivo modelo.php

En el modelo de vista cargamos todos los archivos que se cargarán para que pueda ver el usuario cuando navegue en nuestra aplicación.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <!--Bootstrap para hacer responsive-->
    <meta name="viewport" content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-+0n8XvW+JdUcB+ZlDyE8LwCgA6XfTzIaOpeoHqB+PZDnQ+q7Pj0+Zu+qOZq4JZ&lt;!-->" crossorigin="anonymous">
    <link rel="stylesheet" href=<?php echo constant('URL'); ?>public/css/default.css">
    <title>CAFE-BAR TLAXIACO</title>
</head>
<body>

    <div class="contenedor">

        <div id="header">
            <ul>
                <li><a>CAFE-BAR</a><a href=<?php echo constant('URL'); ?>login">INICIAR SESIÓN</a></li>
            </ul>
        </div>

        <div id="mapa">
            <iframe src=
"https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d10776.272084602226!2d-97.6844825800015!3d17.26783805
width="1350" height="600" style="border:0;" allowfullscreen="" loading="lazy"></iframe>
        </div>
        <!--Script para hacer responsive la aplicación con CSS-->
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous">
        <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js" integrity="sha384-IQsoaAHFcsXkynj8SvKZCkZP+u1Wuq6F7I4geZ3HkfQD6ZB8vZlqDFqfDZD9OOG&lt;!-->" crossorigin="anonymous">
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/js/bootstrap.min.js" integrity="sha384-6ZR8s/2eCqAC11rhTEzZKUu4eXwvDZPZC9eXqZcR2Gq0lWlDUZM7DZq0rDZD9OOG&lt;!-->" crossorigin="anonymous">
    </body>
</html>
```

Ilustración 30 Archivo de vista.php

Por ultimo realizamos prueba de nuestra aplicación que ya está en la web.

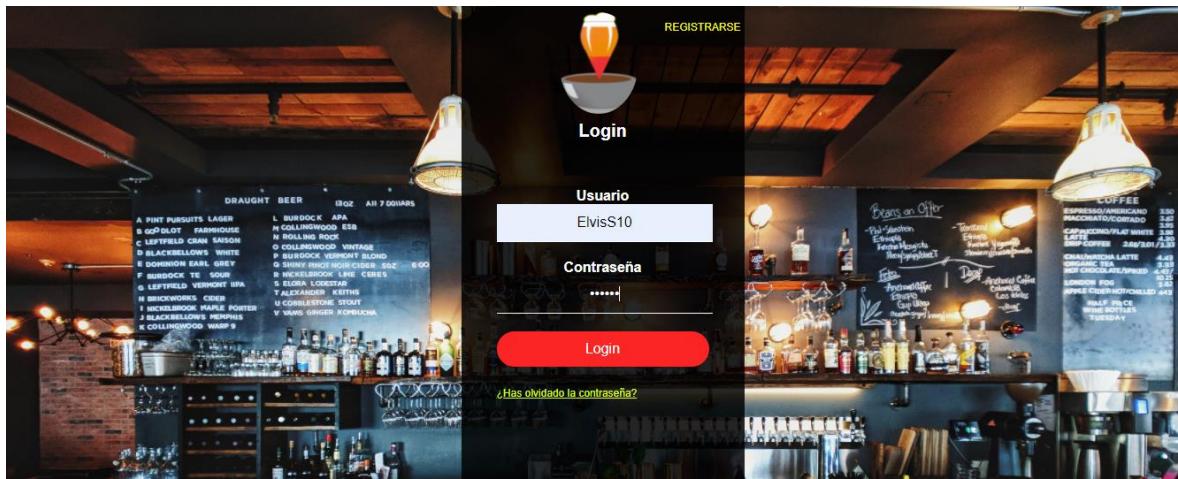


Ilustración 31 Abrimos aplicación y hacemos logeo.

Como podemos ver al cargar un usuario tenemos acceso a nuestra página principal

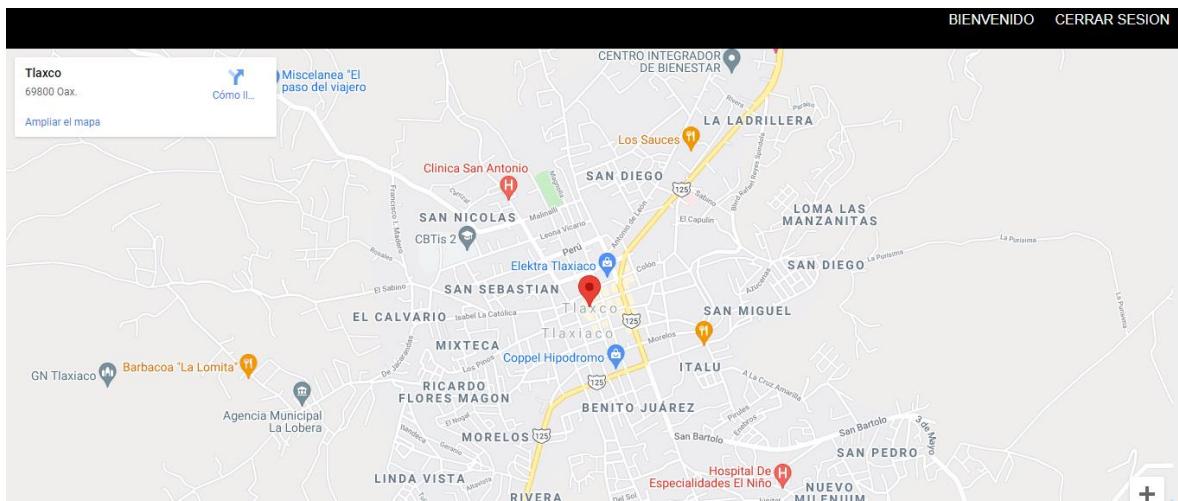


Ilustración 32 Logeo Usuario.

El cliente podrá registrarse o crear un nuevo usuario con características de su negocio.

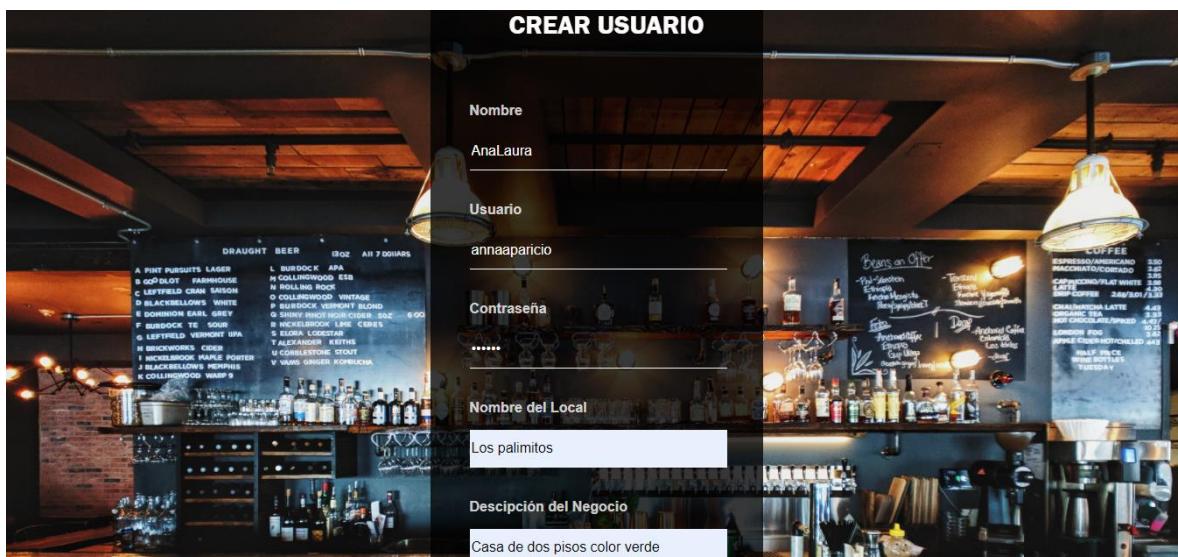


Ilustración 33 Registro de nuevo local.

Como podemos al registrar un nuevo usuario al igual que el local podemos ver que nuestro registro se almaceno correctamente a nuestra base de datos que tenemos de manara remota en Amazonaws.

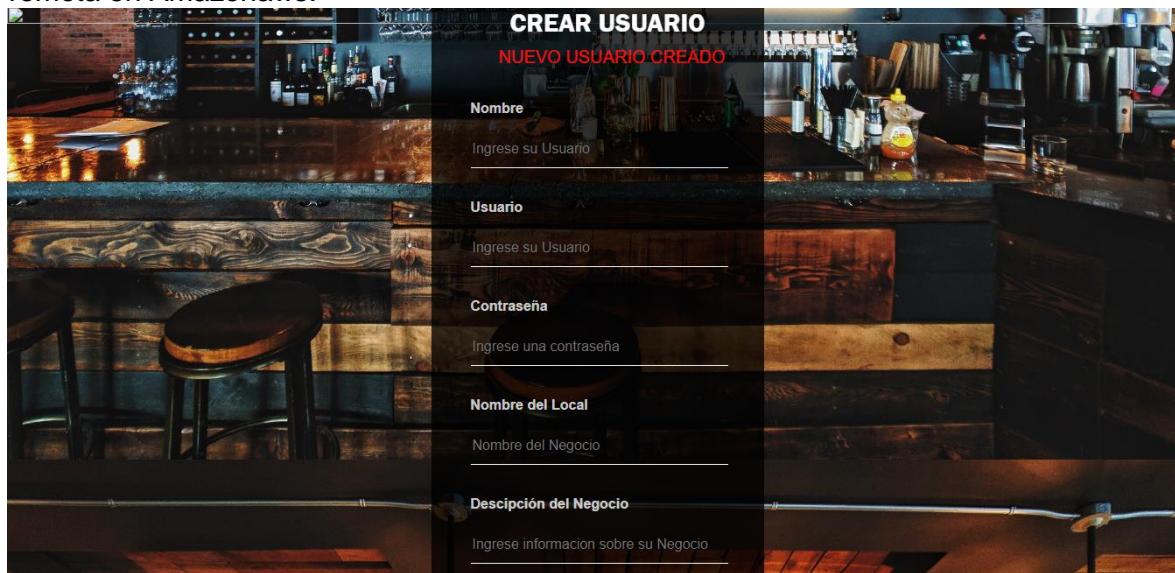


Ilustración 34 Usuario y negocio insertados.

Visualizamos nuestra base de datos y podemos observar que el usuario se almaceno correctamente a nuestra base de datos y con la contraseña encriptada.

Como podemos ver al cargar un usuario tenemos acceso a nuestra página principal

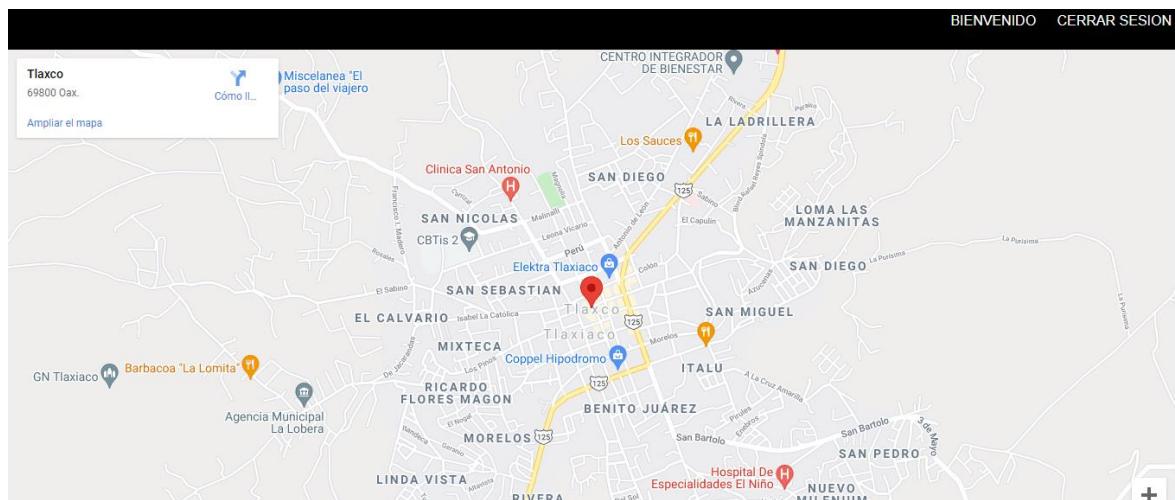


Ilustración 35 Logeo de Usuario.

usuarios		+ Opciones						
		id	nombre	username	seguridad	negocio	descripcion	ubicacion
	Nueva	1	concepcion	madara	fcea920f7412b5da7be0cf42b8c93759	Los palmitos	Casa de dos pisos color verde	Allende S/N - Camino Real #123
	clientes	2	Elvis	ElvisS10	81dc9db52d04dc20036dd8313ed055	Los palmitos	Casa de dos pisos color verde	Allende S/N - Camino Real #123
	dias_de_servicio	3	concepcion	concepcionH	827ccb0eea8a706c4c34a16891f84e7b	Los palmitos	Casa de dos pisos color verde	Allende S/N - Camino Real #123
	Infousuario	4	AnaLaura	annaaparicio	e10adc3949ba59abbe56e057f20f883e	Los palmitos	Casa de dos pisos color verde	Allende S/N - Camino Real #123
	locales							
	productos							
	tipos_cerveza							

Ilustración 36 Base de datos en linea.

Sprint #2

En nuestro sprint #2 hemos mejorado y trabajado en partes del diseño y la mejora de aplicación tanto en el Front-End y en el Back-End. Es por eso que retomamos el Scrum para medir los tiempos de rediseño y codificación para la aplicación.

Front-End

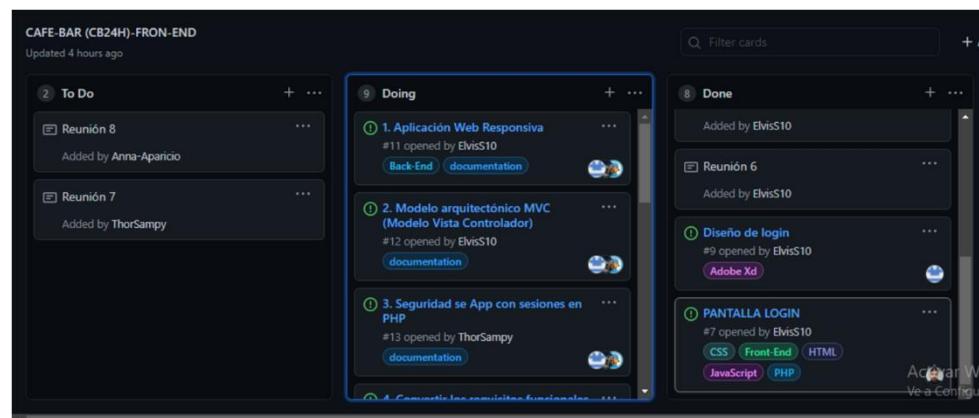


Ilustración 37 Retomamos el desarrollo (1-2-3).

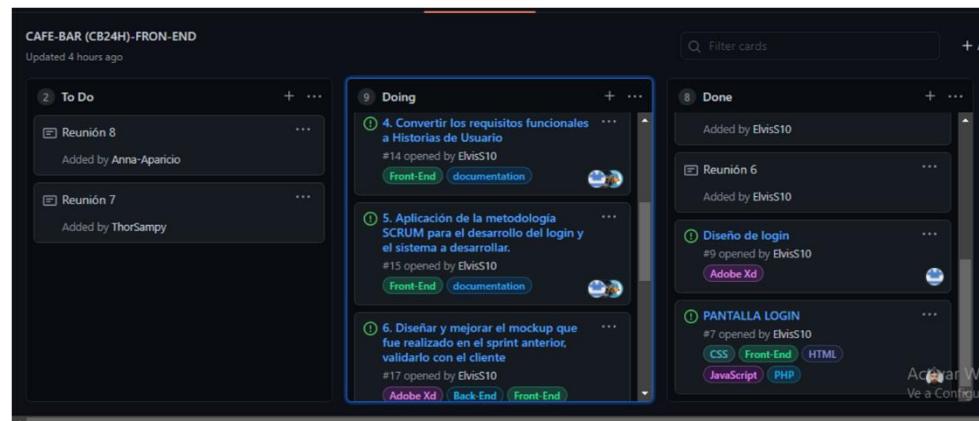


Ilustración 38 Retomamos el desarrollo (4-5-6).



Ilustración 39 Retomamos el desarrollo (7-8-9).

En el sprint #2 retomamos el Scrum del Back-End para reestructurar el código y mejorar la codificación de nuestra aplicación.

Back-End



Ilustración 40 Retomamos el desarrollo (1-2-3-4)

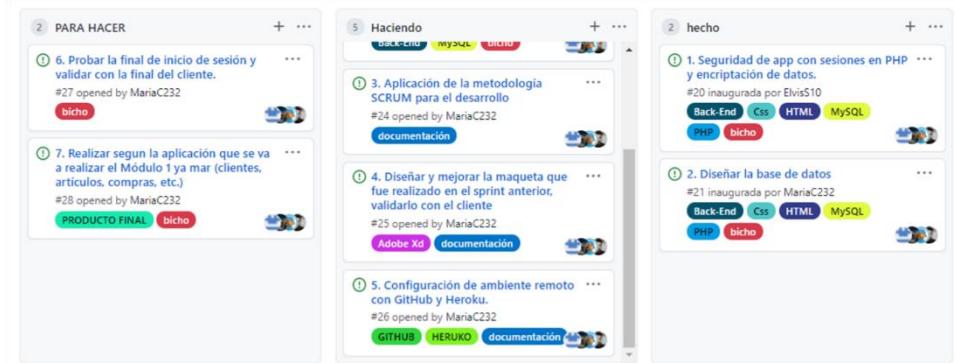


Ilustración 41 Retomamos el desarrollo de (3-4-5)

Diseño de la aplicación final.

Se rediseño la aplicación por lo que nuestra página de inicio estará de la siguiente manera cuando el usuario acceda a nuestro link.

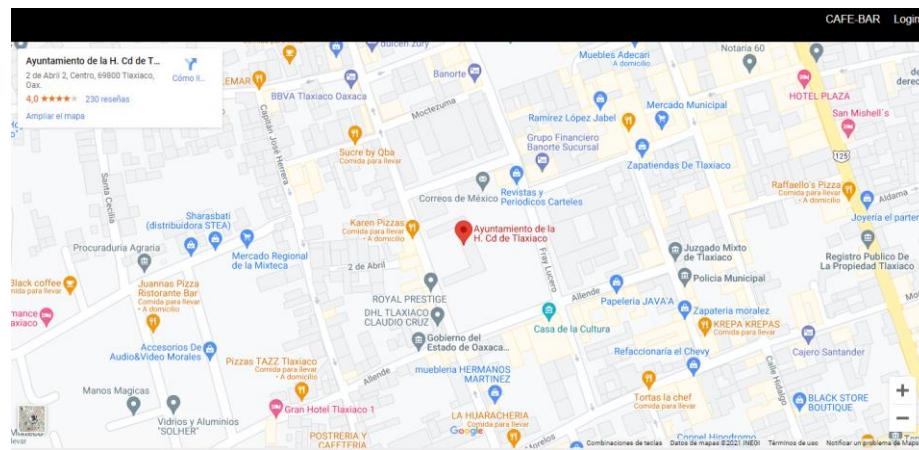


Ilustración 42 Inicio de la aplicación.

Al login solo le hizo un pequeño ajusto en la parte de redirección al registro.

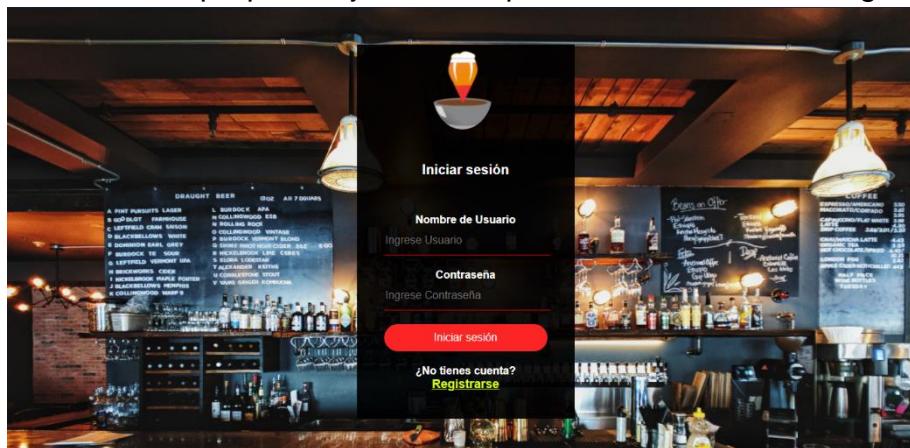


Ilustración 43 Login.

Por ultimo a la venta de registrar se le modifico ya que solo queremos que el usuario registre un nombre de usuario y contraseña.

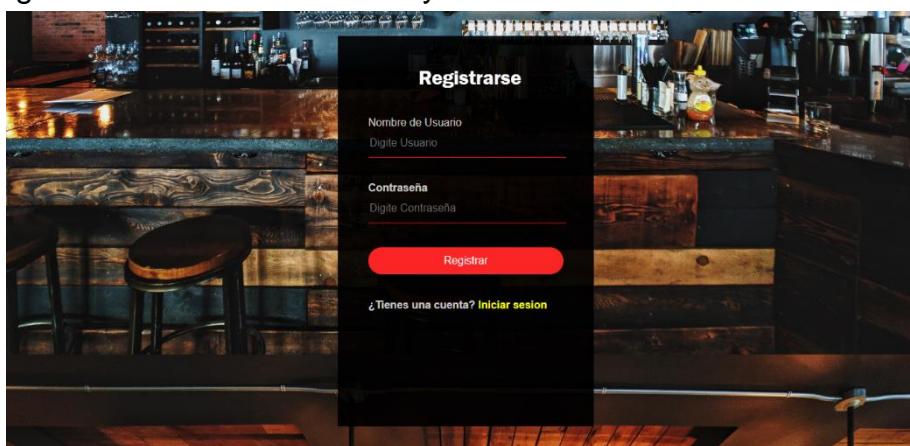


Ilustración 44 Vista de Registro.

La ventana principal después del logeo será la siguiente y como de inicio cargara la vista home o Dashboard.

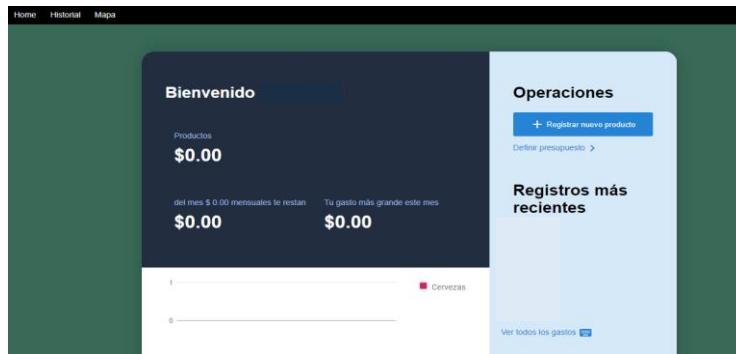


Ilustración 45 Vista Home

Contaremos con una sección de Historial donde el usuario podrá ver los productos que registre para su local.

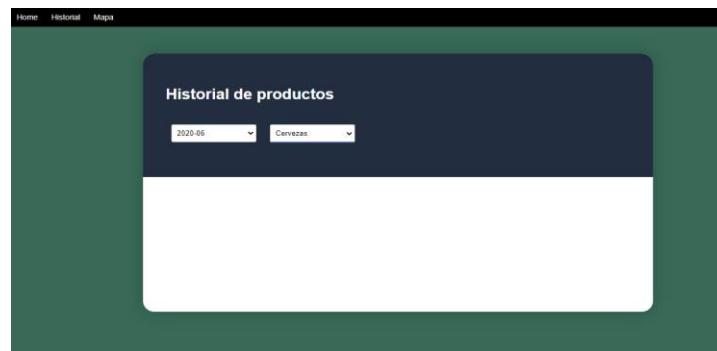


Ilustración 46 Sección de Historial.

En nuestra siguiente sección será la parte donde el usuario fijara el local y de donde se encuentra el negocio.

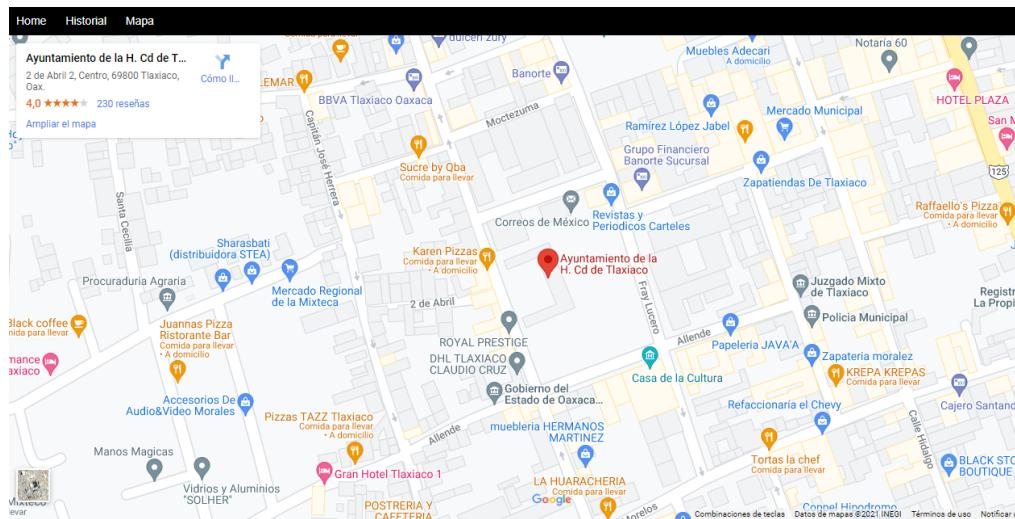


Ilustración 47 Mapa de la aplicación.

Tendremos un panel para que el usuario pueda poner definir un nombre y una foto de perfil para que sea más amigable su instancia en la aplicación.

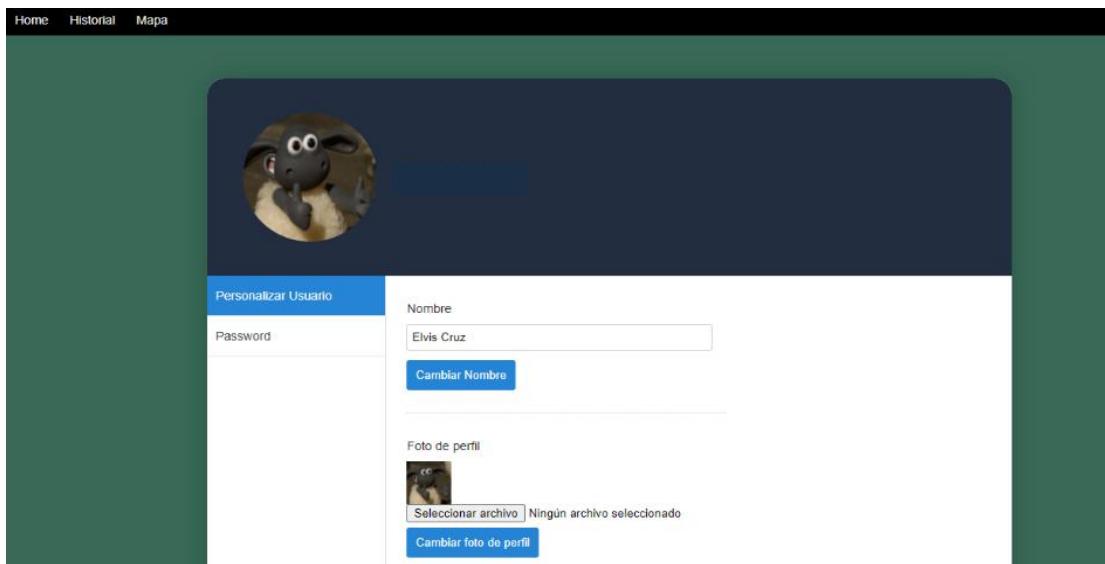


Ilustración 48 Usuario y contraseña

De igual manera tendremos una sección la cual le permitirá al usuario cambiar su contraseña por si no quiere seguir con el que se registró inicialmente.

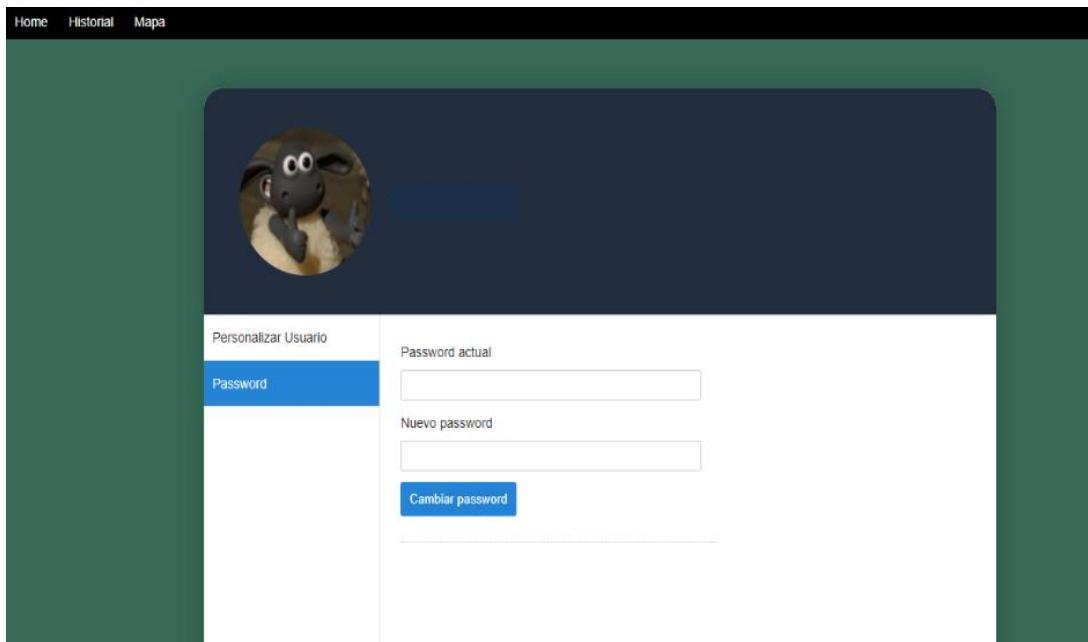


Ilustración 49 Actualizar contraseña.

Codificación de Front-End

En la parte de front-end codificamos y diseñamos nuestras páginas web como se muestra a continuación.

```
index.php ...main x signup.php index.php ...dashboard mapa.php index.php ...expenses index.php ...user
views > main > index.php > @html > @body > @div.contenedor > @div#mapa

4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE-edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-wEemIVmKuiNpc+IOBJ1
8   <link rel="stylesheet" href="https://php echo constant('URL');?>public/css/main.css">
9   <title>Coffee-Bar TLAXIACO</title>
10
11 </head>
12 <body>
13
14 <div class="contenedor">
15
16   <div id="header">
17     <ul>
18       <li><a href="#">CAFE-BAR</a><a href="<?php echo constant('URL'); ?>login">login</a></li>
19       <li><?php
20         $this->showMessages();
21       </?></li>
22     </ul>
23   </div>
24
25   <div id="mapa">
26
27     <iframe src="https://www.google.com/maps/embed?pb=[...]">
28   </div>
29
30   <!-- Script para hacer responsive la aplicacion con css-->
31   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-p3fHUUss3wqzftoSuAdvJ+o
32   <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js" integrity="sha384-IQsLX1SP1fPhsWubn5LC7D9bXGd
33   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/js/bootstrap.min.js" integrity="sha384-1ipyfhUtxLX2ZSbn2FgnHNAK0ahV/OK
34
35 </body>
36 </html>
```

Ilustración 50 Página principal.

```
index.php ...main index.php ...login ✘ signup.php index.php ...dashboard mapa.php index.php ...expenses index.php ...user
views > login > index.php > HTML > body > div.loginbox > form
 6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
 7   <title>Login</title>
 8   <link rel="stylesheet" type="text/css" href="?php echo constant('URL');?>public/css/login.css">
 9
10  </head>
11  <body>
12
13    <?php $this->showMessages();?>
14    <div class="container">
15      
16    </div>
17
18    <div class="loginbox">
19      <h3>Iniciar sesión</h3>
20      
21      <form action="?php echo constant('URL');?>login/authenticate" method="POST">
22
23        <div><?php if(isset($this->errorMessage)) { $this->errorMessage : '' ?></div>
24        <p>
25          <label for="password">Contraseña</label>
26          <input type="password" name="password" id="password" autocomplete="off" placeholder="Ingrese Contraseña">
27        </p>
28        <p>
29          <input type="submit" value="Iniciar sesión" />
30        </p>
31        <p>
32          No tienes cuenta? <a href="?php echo constant('URL');?>signup">Registrarse</a>
33        </p>
34      </form>
35    </div>
36  </body>
37 </html>
```

Ilustración 51 Página de login.

```
index.php ...main signup.php x index.php ...dashboard mapa.php index.php ...expenses index.php ...user
views > login > signup.php > @html > body > div.loginbox > form > p > input#username
views > login > signup.php > @html > body > div.loginbox > form > p > input#username
12 <?php $this->showMessages();?>
13
14
15 <div class="loginbox">
16
17   <form action="php echo constant('URL'); ?/signup/newUser" method="POST">
18
19     <h1 class="animate__animated animate__backInLeft">Registrarse</h1>
20
21     <p>Nombre</p>
22     <input type="text" name="name" placeholder="Ingrese nombre" required>
23
24     <p>      <label for="username">Nombre de Usuario</label>
25     <input type="text" name="username" id="username" placeholder="Digite Usuario" required>
26
27     </p>
28
29     <label for="password">Contraseña</label>
30     <input type="password" name="password" id="password" placeholder="Digite Contraseña" required>
31
32     <p>
33       <input type="submit" value="Registrar" />
34     </p>
35
36     <p>      <a href="php echo constant('URL'); ?/login">Iniciar sesion</a>
37   </p>
38
39 </div>
40 </body>
41 </html>
```

Ilustración 52 Página de registro

Codificación Back-End

Tenemos todos los componentes para el funcionamiento de SQL así como las vistas en nuestras páginas web.

```
function render()
{
    error_log('Dashboard::render -> Carga el index de Dashboard');
    $expensesModel = new ExpensesModel();
    $expenses = $this->getExpenses();
    $totalThisMonth = $expensesModel->getTotalAmountThisMonth($this->user->getId());
    $maxExpensesThisMonth = $expensesModel->getMaxExpensesThisMonth($this->user->getId());
    $categories = $this->getCategories();

    $this->view->render('dashboard/index', [
        'user' => $this->user,
        'expenses' => $expenses,
        'totalAmountThisMonth' => $totalThisMonth,
        'maxExpensesThisMonth' => $maxExpensesThisMonth,
        'categories' => $categories
    ]);
}

private function getExpenses($n = 0)
{
    if ($n < 0) return NULL;
    //error_log("Dashboard::getExpenses() id = " . $this->user->getId());

    $expenses = new ExpensesModel();
    return $expenses->getByUserIdAndLimit($this->user->getId(), $n);
}
```

Ilustración 53 Controlador principal para la vista Home.

```
define('URL', 'http://localhost/coffe-bar/');
define('HOST', 'localhost');
define('DBNAME', 'coffe_bar');
define('USER', 'root');
define('PASSWORD', '');
define('CHARSET', 'utf8mb4');
```

Ilustración 54 Conexión a la base de datos.

```
<?php
require_once 'models/expensesmodel.php';
require_once 'models/categoriesmodel.php';

class Dashboard extends SessionController{

    private $user;

    function __construct(){
        parent::__construct();
        $this->user = $this->getUserSessionData();
        error_log('Dashboard::construct -> Carga el index de Dashboard');
    }

    function render()
    {
        error_log('Dashboard::render -> Carga el index de Dashboard');
        $expensesModel = new ExpensesModel();
        $expenses = $this->getExpenses();
        $totalThisMonth = $expensesModel->getTotalAmountThisMonth($this->user->getId());
        $maxExpensesThisMonth = $expensesModel->getMaxExpensesThisMonth($this->user->getId());
        $categories = $this->getCategories();

        $this->view->render('dashboard/index', [
            'user' => $this->user,
            'expenses' => $expenses,
            'totalAmountThisMonth' => $totalThisMonth,
            'maxExpensesThisMonth' => $maxExpensesThisMonth,
            'categories' => $categories
        ]);
    }
}
```

Ilustración 55 Controlador de Historial.

En nuestra siguiente imagen es donde se realiza toda la parte de autentificación de los usuarios al momento de logearse con el nombre de usuario y contraseña.

```

<?php
require_once 'models/usermodel.php';

class LoginModel extends Model{
    public function __construct(){
        Parent::__construct();
    }

    public function login($username, $password){
        // insertar datos en la BD
        //error_Log("Login: inicio");
        try{
            //$query = $this->db->connect()->prepare('SELECT * FROM users WHERE username = :username');
            $query = $this->prepare('SELECT * FROM users WHERE username = :username');
            $query->execute(['username' => $username]);

            if($query->rowCount() == 1){
                $item = $query->fetch(PDO::FETCH_ASSOC);

                $user = new UserModel();
                $user->from($item);

                //error_Log('Login: user id '.$user->getId());
            }

            if(password_verify($password, $user->getPassword())){
                error_log('LoginModel::login->success');
                //return ['id' => $item['id'], 'username' => $item['username'], 'role' => $item['role']];
                return $user;
                //return $user->getId();
            }else{
                error_log('LoginModel::login->password NO ES VALIDA');
            }
        }catch(PDOException $e){
            error_log($e);
        }
    }
}

```

Ilustración 56 Archivo de autentificación de usuarios.

Contamos con un archivo Js la cual es donde definimos las funciones y operaciones que se realizaran como en nuestro caso al momento que se registre, el usuario no exista etc. Son funciones para ser más intuitivo la aplicación.

```

const btnExpense = document.querySelector('#new-expense');

btnExpense.addEventListener('click', async e =>{
    const background = document.createElement('div');
    const panel = document.createElement('div');
    const titlebar = document.createElement('div');
    const closeButton = document.createElement('a');
    const closeButtonText = document.createElement('i');
    const ajaxContent = document.createElement('div');

    background.setAttribute('id', 'background-container');
    panel.setAttribute('id', 'panel-container');
    titlebar.setAttribute('id', 'title-bar-container');
    closeButton.setAttribute('class', 'close-button');
    //closeButton.setAttribute('href', '#');
    closeButtonText.setAttribute('class', 'material-icons');
    ajaxContent.setAttribute('id', 'ajax-content');

    background.appendChild(panel);
    panel.appendChild(titlebar);
    panel.appendChild(ajaxContent);
    titlebar.appendChild(closeButton);
    closeButton.appendChild(closeButtonText);
    closeButtonText.appendChild(document.createTextNode('close'));
    document.querySelector('#main-container').appendChild(background);

    closeButton.addEventListener('click', e =>{
        background.remove();
    });
})

```

Ilustración 57 Archivo Js.

Tenemos un archivo muy importante la cual nos permitió hacer muchas correcciones a nuestra aplicación siendo de mucha ayuda ya que es como hemos encontrado todos los bugs que íbamos generando a través del desarrollo de nuestra aplicación, y cómo podemos observar en la siguiente imagen obtuvimos un total de **32,628** errores. La cual logreamos completar y hacer que la aplicación final no tuviera fallas cuando se le entregue al usuario final.

```

File Edit Selection View Go Run Terminal Help
php-error.log - coffe-bar - Visual Studio Code

EXPLORER OPEN EDITORS php-error.log
COFFE-BAR.css arrow.png dash.css defa.css error.css expense.css history.css home.css loginn.css main.css signup.css user.css
imagenes bar.jpg cb.png desktop.ini
img photos 3ba056f17d24...
js admin.js dashboard.js
views .htaccess index.php
php-error.log

32610 [13-Jun-2021 08:44:09] europe/Berlin] SessionController::getUserSessionData -> elviss10
32611 [13-Jun-2021 08:50:45] Europe/Berlin] inicio de aplicacion!
32612 [13-Jun-2021 08:50:48] Europe/Berlin] SessionController::validateSession
32613 [13-Jun-2021 08:50:48] Europe/Berlin] SessionController::getUserSessionData -> elviss10
32614 [13-Jun-2021 08:50:48] Europe/Berlin] SessionController::getCurrentPage -> user
32615 [13-Jun-2021 08:50:48] Europe/Berlin] SessionController::getCurrentPage -> user
32616 [13-Jun-2021 08:50:48] Europe/Berlin] SessionController::getUserSessionData -> elviss10
32617 [13-Jun-2021 09:03:43] Europe/Berlin] inicio de aplicacion!
32618 [13-Jun-2021 09:03:44] Europe/Berlin] SessionController::validateSession
32619 [13-Jun-2021 09:03:45] Europe/Berlin] SessionController::getUserSessionData -> elviss10
32620 [13-Jun-2021 09:03:45] Europe/Berlin] SessionController::getCurrentPage -> logout
32621 [13-Jun-2021 09:03:45] Europe/Berlin] SessionController::getCurrentPage -> logout
32622 [13-Jun-2021 09:03:45] Europe/Berlin] SessionController::redirectDefaultSiteByRole -> role = user
32623 [13-Jun-2021 09:03:45] Europe/Berlin] SessionController::redirectDefaultSiteByRole -> role = expenses/mapa
32624 [13-Jun-2021 09:03:45] Europe/Berlin] inicio de aplicacion!
32625 [13-Jun-2021 09:03:45] Europe/Berlin] SessionController::validateSession
32626 [13-Jun-2021 09:03:45] Europe/Berlin] SessionController::getCurrentPage -> login
32627 [13-Jun-2021 09:03:45] Europe/Berlin] Login::__construct -> Inicio login
32628 [13-Jun-2021 09:03:45] Europe/Berlin] Login::__render -> Carga el index de login
32629

```

Ilustración 58 Archivos para Bugs.

Resultado final

Hemos pruebas finales para comprobar si es funcional la aplicación. Hacemos un logeo para comprobar funcione nuestra aplicación y al momento de hacer el logeo nuestra sección de **home** nos carga exitosamente.



Ilustración 59 Vista Home.

Si nos dirigimos a la sección de **Historial** nos carga perfectamente y nos muestra información que hemos agregado y comprobamos que los datos se cargan perfectamente que están almacenados en nuestra base de datos.

The screenshot shows a dark-themed web application interface. At the top, there are three navigation buttons: 'Home', 'Historial' (which is highlighted with a red border), and 'Mapa'. In the top right corner, it says 'Elvis Cruz' with a small profile icon. Below the navigation, the title 'Historial de productos' is displayed. Underneath the title are two red-bordered buttons: 'Ver todas las fechas' and 'Ver todas las categorías'. A large table follows, also enclosed in a red border. The table has columns: 'Título', 'Productos', 'Fecha', 'Cantidad', and 'Acciones'. The data in the table is as follows:

Título	Productos	Fecha	Cantidad	Acciones
CORONA	Cervezas	2020-06-01	\$15	<button>Eliminar</button>
WISKEY	Cervezas	2020-06-03	\$250	<button>Eliminar</button>
VODKA	Cervezas	2020-06-03	\$200	<button>Eliminar</button>
Comidas pollos	Comidas	2021-06-13	\$66	<button>Eliminar</button>

Ilustración 60 Sección de **Historial**.

Si nos dirigimos a la parte de **Mapa** podemos ver que nuestro mapa nos carga exitosamente que es donde el usuario podrá definir su ubicación del local.

Nota: En la parte del mapa esta en desarrollo y en investigación para su implementación.

The screenshot shows a map of the town of Tlaxiaco, Oaxaca. The map includes street names like '2 de Abril', 'Fray Luero', 'Allende', 'Calle Hidalgo', and 'Avenida Constitución'. Various local businesses and landmarks are marked with pins, including 'Ayuntamiento de la H. Cd de Tlaxiaco', 'BBVA Tlaxiaco Oaxaca', 'Mercado Municipal', 'HOTEL PLAZA', 'Karen Pizzas', 'Revistas y Periodicos Carteles', 'Juzgado Mixto de Tlaxiaco', 'Policía Municipal', 'Papelaria JAVA'A', 'Zapatería moralez', 'KREPA KREPAS', 'LA HUARACHERIA', 'Refaccionaria el Chevy', 'Tortas la chef', 'BLACK STORE BOUTIQUE', and 'Coppel Hinojosa'. On the left side, there is a sidebar with information about the 'Ayuntamiento de la H. Cd de Tlaxiaco', which has a rating of 4.0 stars and 230 reviews. It also includes links for 'Cómo llegar', 'Ampliar el mapa', and other local business listings like 'Black coffee', 'Juanitas Pizza Ristorante Bar', 'Accesorios De Audio&Video Morales', 'Manos Mágicas', 'Vidrios y Aluminios "SOLHER"', 'POSTERIA Y CAFFETTERIA', 'Gobierno del Estado de Oaxaca...', 'muebleria HERMANOS MARTINEZ', 'LA HUARACHERIA', 'Coppel Hinojosa', and 'BLACK STORE BOUTIQUE'.

Ilustración 61 Sección **Mapa**.

En nuestro panel de usuario que es el perfil es donde hemos definido un nombre y hemos cargado una foto la cual este se carga exitosamente al igual que el nombre y si podemos observar en la parte inferior izquierda se nos muestra el nombre y la foto de perfil.

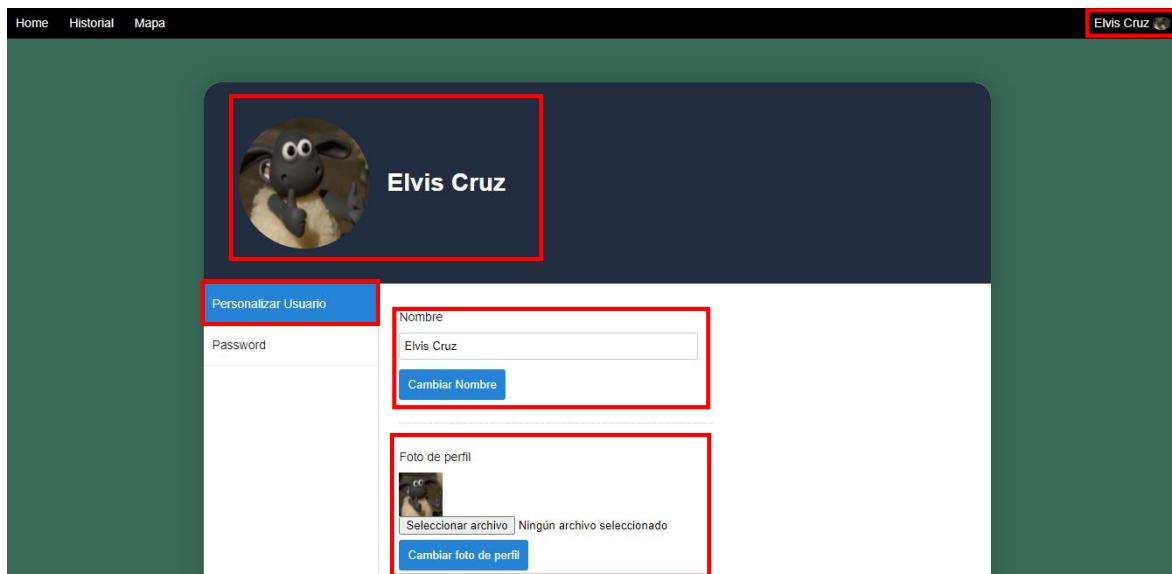


Ilustración 62 Nombre y foto cargados exitosamente.

En nuestra sección de Password nos carga bien nuestra interfaz para que el usuario cuando desee pueda modificar su contraseña si lo desea.

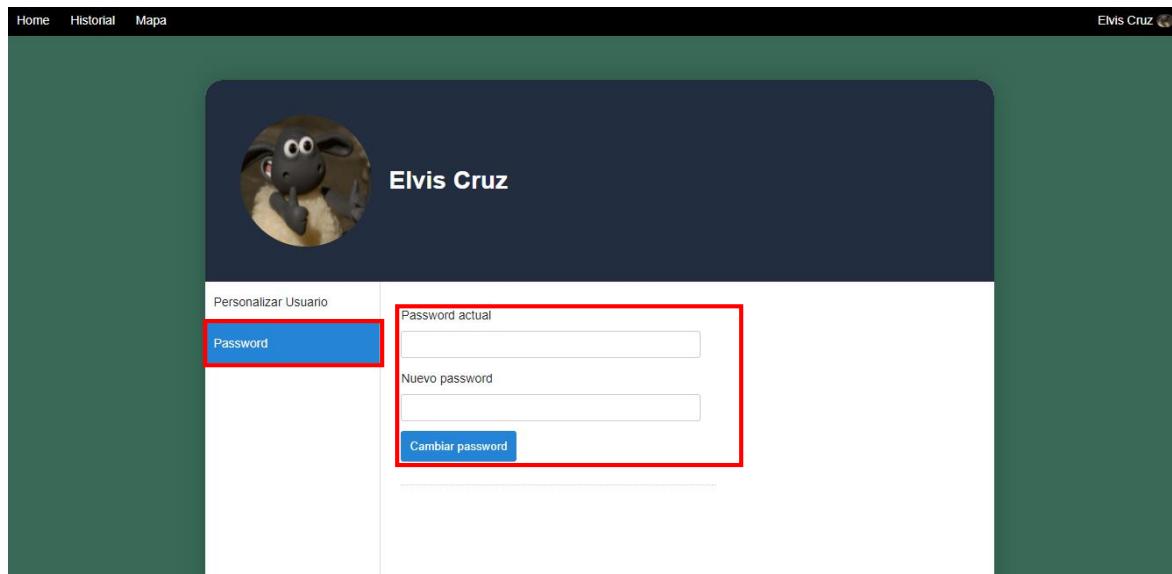


Ilustración 63 Panel de Actualizar la contraseña.

Conclusión

Una vez realizado el desarrollo de la documentación sobre nuestra aplicación y la arquitectura del software que lo compondrá para su desarrollo, podemos decir que un buen diseño de interfaz y del alcance que este tendrá. En el diseño de la aplicación se realizó una buena interfaz con funcionalidades amigables y que al usuario le sea agradable para que este sea llamativo y el usuario pueda y quiera volver a utilizar nuestra aplicación a futuro. De igual manera hemos diseñado la estructura de base de datos que es la base más importante de todo software, la cual la hemos diseñado en el software de architec en el cual se definieron las variables más importantes con las cuales trabajara nuestra aplicación.

Cabe recalcar que este es un prototipo y que primero se llevará a cabo en una plataforma web, con la cual se trabajará con el prototipo actual y de igual manera con la estructura de base de datos que hemos diseñado para la aplicación, conforme avancemos en la materia lo iremos actualizando con mejores funcionalidades que para el usuario se le haga mejor y pueda contar con la tecnología actual.

El proyecto se estará trabajando desde un repositorio de forma local donde se estarán subiendo los códigos con la que se desarrollara la aplicación, código **php** con el que se estará trabajando y es un lenguaje de alto nivel, de igual manera con el modelo vista controlador (**MVC**), **HTML**, **CSS**, **H5** y demás componentes para el desarrollo de la aplicación, como último punto se estará subiendo la aplicación a un servidor para encontrarlo en línea, el servidor donde se cargara la aplicación será **HERUKO**.

REFERENCIA

- Bahit, E. (2011). Poo y mvc en php. *El paradigma de la Programación*.
- Zulian, E. R. (2011). *Implementación de un framework para el desarrollo de aplicaciones web utilizando patrones de diseño y arquitectura MVC/REST* (Doctoral dissertation, Universidad de Belgrano. Facultad de Tecnología Informática.)
- González, Y. D., & Romero, Y. F. (2012). Patrón Modelo-Vista-Controlador. *Telemática*, 11(1), 47-57.
- FERNÁNDEZ, L. (2006). Arquitectura de software. *Software Guru*, 2(3), 40-45.
- Celma Giménez, M., Casamayor Ródenas, J. C., & Mota Herranz, L. (2003). Bases de datos relacionales.
- Servidor para repositorios de Proyectos*. (s. f.). GitHub. Recuperado 15 de junio de 2021, de <https://github.com/>
- Servidor Web*. (s. f.). Heruko. Recuperado 15 de junio de 2021, de <https://www.heroku.com/>